SourceGate™ II *Lite*

for

User's Guide

Version 1.1.0

June 20, 1994

Huntsville Microsystems, Inc. © 1994

The contents of this document are believed to be accurate. However, Huntsville Microsystems, Inc. assumes no responsibility for the use of this document and issues no warranties, implied or otherwise, on the contents herein. Furthermore, Huntsville Microsystems, Inc. reserves the right to revise this publication and to make changes to the contents, without obligation to notify any persons of such revisions or changes.

Copyright © 1994 by Huntsville Microsystems, Inc.

SourceGate is a registered trademark of Huntsville Microsystems, Inc.
Hercules is a registered trademark of Hercules Computer Technology.
Pentium is a registered trademark of Intel Corporation.
Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation.

This document was created entirely with AmiPro, which is a trademark of Lotus Development Corporation.

TABLE OF CONTENTS

Chapter 1 - Getting Started 1

Checking Your Package 1

Disks 1

Documentation 1

Hardware 1

System Requirements 1

Hardware 1

Software 2

Memory 2

Disk Space 2

Installing SourceGate II 2

Starting SourceGate II 2

Emulator To Computer Connection 2

Emulator To Target System Connection 2

Starting SourceGate II From Program Manager 2

Starting SourceGate II From File Manager 3

Command Line Options 3

Environment Variables 5

Chapter 2 - General Overview 6

Becoming Familiar With The SourceGate II Window 6

Control Menu Boxes 6

Menu Bar 6

Title Bar 7

Restore Button 7

Maximize Button 7

Minimize Button 7

Scroll Bars (Horizontal and Vertical) 7

Scroll Boxes (Horizontal and Vertical) 7

TABLE OF CONTENTS (Cont.)

SourceGate II Menu Bar 7
The 'File' Menu 7
Download Hex File 7
Upload Hex File 7
Execute 7
Record 8
Log 8
Print 8
Printer Setup 8
Exit SourceGate 8
The 'Edit' Menu 8
The 'Setup' Menu 8
Event 8
Breakpoint 8
Trace Trigger 8
The 'Display' Menu 8
Command Window 8
Trace 9
Watch 9
Watch In Real Time 9
Registers 9
Memory 9
Stack Trace 9
Run Status 9
The 'Source' Menu 9
Run 9
Go From/To 9
Step 9
Stop 9
Reset Communications 10
Reset Emulator 10

TABLE OF CONTENTS (Cont.)

Symbols 10
Modules 10
Assemble 10
VASM 10
Tracking Assembly 10
The 'Config' Menu 10
Memory 10
Source Code Interface 10
The 'Windows' Menu 11
Next Window 11
Previous Window 11
Cascade Windows 11
Close 11
Close All 11
Open Window List 11
The 'Help' Menu 11
On Context 11
On Window 11
Keyboard 11
Index 11
How Do I... 12
Help On Help 12
Search 12
About 12
Maximizing/Minimizing Windows 13
Window Manipulation 15
Changing To The Next Window 15
Changing To The Previous Window 16
Choosing From The Open Window List 16
Cascading All Open Windows 16

TABLE OF CONTENTS (Cont.)

Closing The Current Active Window 16
Closing All Open Windows 16
Defining Your Printer 16
Working With Symbols 17
Using Help 17
Context Sensitive Help 18
Active Window Help 18
Help Menu 18
On Context 18
On Window 18
Keyboard 18
Index 18
How Do I... 18
Help On Help 19
Search 19
About 19
SourceGate II Help Window 20
Help Window Buttons 20
Contents 20
Search 20
Back 21
History 21
Help Window Menus 21
File 21
Edit 21
Bookmark 21
Help 21
Cross-References 21
Exiting SourceGate II 22
Automating SourceGate II 23
Accelerator Keys 24

TABLE OF CONTENTS (Cont.)

Chapter 3 - The Emulation Interface 25
Defining The Emulator's Configuration Window 25
Terminal Type Configuration Windows 25
Terminal Type Configuration Window Fields 27
Clock Field 27
Target Reset Field 27
Target Halt Field 27
Rom Module Boot Field 28
Register Reset Values Fields 28
Terminal Type Configuration Window Memory Map Fields 28
Activator Fields 28
CSBOOT, CS0-10 Field 29
CS0-3 Field 29
Size Field 29
Base Field 30
FC2-0 Field 30
FC3-0 Field 30
Width Field 30
RO and RW Fields 30
Memory Used and Memory Left Fields 31
Graphical Configuration Windows 31
Source Code Interface Configuration 32
Working With Object Files 33
Downloading An Object File 33
File Down Load Options Window Fields 34
'Target Offset' Field 34
'AUX Port Command' Field 35
'Records To Load' Fields 35
'Port Selection' Fields 35
'Symbol Options' Fields 35
File Down Load Options Window Buttons 36

TABLE OF CONTENTS (Cont.)

Uploading An Object File 37
File Up Load Options Window Fields 37
Records To Dump 37
Target Starting Address 37
Target Ending Address 37
File Up Load Options Window Buttons 38
Using The Command Window 39
Opening The Command Window 39
Command Window Buttons 40
Entering Command Window Commands 40
Command Line Overrides 41
S (Supervisor Mode) 41
U (User Mode) 41
P (Program Mode) 41
D (Data Mode) 41
L (Long Word Mode) 41
W (Word Mode) 41
B (Byte Mode) 41
T (Target System) 41
R (Repeat) 42
Override Example 42
Using Symbols In The Command Window 43
Using Module Commands In The Command Window 44
:=A [module] Assembly Mode Command 44
:=C [module] Clear Symbol Table Command 44
:=D [module] Display Current Modes Command 44
:=M [module] Mixed Mode Command 44
:=S [module] Source Mode Command 45
Creating Automated Command Files From The Command Window 45

TABLE OF CONTENTS (Cont.)

Available Command Window Commands 46
A (In-Line Assembly) Command 48
BM (Byte Mode) Command 49
BSD(+/-) (Tri-State Data Bus Buffers During DMA) Command 50
BTE (Set 8-Bit Boot Chip Select) Command 51
C (Compare Memory Block) Command 52
CM (Show Status Of Control Board Memory) Command 54
CONFIG (Recall Factory Configuration) Command 55
CONFIGT (Recall Factory Configuration For Target) Command 56
CRL (Load Configuration Registers) Command 57
D (Dump Memory Block) Command 58
DDP (Dump Dual Port Memory Block) Command 59
DM (Data Memory Mode) Command 60
E/EN (Examine/Change Memory) Command 61
EDP/EDPN (Examine/Change Dual Port Memory) Command 62
EXP (Examine/Change Peripheral) Command 63
EXR (Examine/Change Peripheral Register) Command 64
F (Fill Memory Block) Command 65
FDP (Fill Dual Port Memory Block) Command 66
G (Begin Real-Time Emulation) Command 67
HLD (Hex Listing Disabled) Command 68
HLE (Hex Listing Enabled) Command 69
I/IN (Input) Command 70
L (List Disassembled Code) Command 71
LM (Long Word Display Mode) Command 72
M (Move Memory Block) Command 73
MMC (Check Configuration Menu Setup) Command 74
MT/MTL (Test Block Of Memory) Command 75
O/ON (Output) Command 76
PM (Program Memory Mode) Command 77
RD/RDP (Read Object File Into Memory) Command 78

TABLE OF CONTENTS (Cont.)

RD(%) (Read Configuration File) Command 80
RD(!) (Read Firmware File Into Flash ROM) Command 81
RP (Reset Processor, Continue Emulation) Command 82
RS (Reset Processor, Stop Emulation) Command 83
SH (Show Status For SHD/SHE) Command 84
SHD (Serial Handshake Disabled) Command 85
SHE (Serial Handshake Enabled) Command 86
SM (Supervisor Memory Mode) Command 87
SP (Stop Emulation) Command 88
SR/SRN (Search Block Of Memory) Command 89
SS (Single Step) Command 91
ST (Show Emulator Status) Command 94
UM (User Memory Mode) Command 95
VASM (View Assembly) Command 96
VER (Display Firmware Version) Command 97
W (Write Memory Contents To File) Command 98
W(%) (Write Configuration File) Command 100
WM (Word Display Mode) Command 101
X (Examine/Change Registers) Command 102
Setting Up Events 104
Opening The Event Windows 104
Event Window Fields 105
Event Address Field 105
Event Address Range Field 105
Event Symbol Entry Field 105
Event Range Active Field 105
Event 'Qualify On Opcode Execution' Field 105
Event 'R/W' Field 106
Event Window Buttons 106
Event Binary Address Window 106
Event Binary Address Window Buttons 107

TABLE OF CONTENTS (Cont.)

Defining Breakpoints and Trace Triggers 108
Opening The Sequence Window 108
Sequence Window Fields 108
Sequence Window 'ON/OFF' Field 109
Sequence Window 'Sequence' Field 109
Sequence Window 'Cycles to Delay After Event' Field 109
Sequence Window Buttons 109
Working With The Trace Window 110
Opening The Trace Window 111
Trace Display Modes 112
Disassembly 113
State 114
Source 115
Disassembly/State 116
Disassembly/Source 117
Disassembly/State/Source 118
Freeze Trace Operation 119
Freezing The Trace 119
Resuming The Trace 119
Searching The Trace 120
68331/68332/68HC16 Lite Trace Search Status Bits 122
68340 Lite Trace Search Status Bits 123
Trace Search Window Buttons 124
Printing The Trace 125
Print Trace Buffer Window Buttons 125
Trace Window Buttons 126
Working With The Watch Window 127
Opening The Watch Window 127
Adding A Variable To The Watch Window 128
Entering Watch Variables Manually From The Watch Window 128

TABLE OF CONTENTS (Cont.)

Using The Watch Entry Window To Enter Watch Variables 130
Adding A Watch Variable Manually 130
Adding A Watch Variable Using Buttons 132
Deleting A Variable From The Watch Window 134
Saving A Watch Window 134
Restoring A Saved Watch Window 134
Watch Window Buttons 134
Watch Window Examples 135
Watching A Structure 135
Watching An Element Of A Structure 136
Watching A One-Dimensional Array 136
Watching A Multi-Dimensional Array 136
Watching Pointers 137
Using Overrides In The Watch Window 137
Working With The Watch In Real Time Window 138
Opening The Watch In Real Time Window 138
Working With The Register Window 139
Opening The Register Window 140
Register Window Contents 140
Program Counter Register 140
Program Counter Description Field 140
Status Register 140
Supervisor Stack Pointer Register 141
User Stack Pointer Register 141
Vector Base Register 141
Data Registers D0-D7 141
Address Registers A0-A7 141
Alternate Function Code Registers 142
Working With The Memory Window 143
Opening The Memory Window 143
Modifying Memory Values 144

TABLE OF CONTENTS (Cont.)

Comparing Memory Blocks 144
Memory Compare Window Buttons 144
Memory Output Window Buttons 145
Filling Memory With A Constant 146
Memory Fill Window Buttons 146
Viewing A Specific Address In Memory 147
Memory Goto Window Buttons 147
Reading/Writing A Specific Memory Location 148
Memory I/O Window Buttons 148
Moving Memory Blocks 149
Memory Move Window Buttons 149
Searching Memory Blocks 150
Memory Search Window Buttons 150
Memory Search Output Window Buttons 151
Testing Memory Blocks 152
Memory Test Window Buttons 152
Memory Output Window Buttons 153
Specifying The Display Format 154
Memory Display Format Window Buttons 154
Memory Window Buttons 155
Working With The Stack Trace Window 156
Opening The Stack Trace Window 156
Stack Trace Window Contents 156
Working With The Run Status Window 158
Opening The Run Status Window 158
Run Status Window Fields 158
Run Status 'Status' Field 158
Run Status Trace Field 159
Run Status Window Buttons 159

TABLE OF CONTENTS (Cont.)

Chapter 4 - The Source Code Interface 160

Using The Code Window 160

Opening A Code Window 161

Code Window Format 162

Setting Code Window Breakpoints 163

Deleting Code Window Breakpoints 163

Listing Code Window Breakpoints 164

Breakpoint List/Delete Breakpoint Window Buttons 165

Finding Subsequent Code Window Breakpoints 165

Finding The Current Program Counter While In A Code Window 165

Running To A Highlighted Code Window Line 166

Single Stepping In A Code Window 166

Searching For A Code Window String 166

Viewing Code Window Assembly 167

Viewing Code Window Symbols 168

Selecting Other Modules From A Code Window 169

Using The VASM Window 170

Using The Tracking Assembly Window 171

Using The Single Step Window 172

Single Step Window Fields 172

Address Field 172

Current Instruction Field 172

Step Continuously Check Box 172

Step Over Subroutines Check Box 172

Quiet Mode Check Box 172

Step Count Field 173

Step Until Field 173

Single Step Window Buttons 174

TABLE OF CONTENTS (Cont.)

Using The Go From/To Window 175

Go From/To Window Fields 175

Go From Address Field 175

Go To Breakpoints Fields 175

Go From/To Window Buttons 176

Appendix A Error Messages 178

Appendix B Using Compilers and Assemblers 183

1 Getting Started

Checking Your Package

This section lists the disks, documentation, and hardware included in your SourceGate II for Windows emulation package. Take time now to check the contents of your package. If your package is not complete, contact HMI.

Disks

Your SourceGate II package should contain one 3.5" disk labeled 'SourceGate II'.

Documentation

Your package should contain the following documentation:

- *How to Use This Manual*
- *SourceGate II Lite for Windows User's Guide*
- *Getting Started With HMI Emulators and SourceGate II*
- *Application Notes*

Hardware

Your package should contain the emulator and interface adapter ordered along with a RS232 serial cable and power supply.

System Requirements

This section describes the hardware, software, memory, and disk space requirements for using SourceGate II on a stand-alone computer or on a network.

Hardware

SourceGate II requires the following hardware:

- *A 286, 386, 486, or Pentium™ based computer certified for use with Windows 3.1*
- *An EGA, VGA, super VGA, or Hercules® graphics card compatible with Windows 3.1*
- *A Windows compatible mouse is recommended but not required.*
- *One 1.44MB 3.5" disk drive*
- *One RS232 serial port*
- *One Centronics compatible parallel port recommended but not required*
- *Hard disk drive with at least 4MB of free disk space*

Software
You must have Microsoft Windows Version 3.1 or higher and DOS Version 3.1 or higher installed on your computer to use SourceGate II.

Memory
SourceGate II requires a minimum of 2MB of extended memory with 4MB recommended.

Disk Space
SourceGate II requires a minimum of 4MB of hard disk space.

Installing SourceGate II
To install SourceGate II, place the disk labeled "SourceGate II" in your 1.44MB 3.5" disk drive, open the Windows File Manager, make the drive where the floppy is located to be the active drive, and run the "SETUP.EXE" program. You should then follow the instructions in the setup program in order to install the SourceGate II files correctly. A Program Group will be created with the SourceGate II icon shown in this group.

Starting SourceGate II
SourceGate II can be started from Windows in several ways. There are also environment variables and command line options that can be defined that influence the way SourceGate II will behave. First, however, you need to connect an emulator to your computer. If you are working with a target system, you must also connect the emulator to the target system. The following sections tell you how to do this.

Emulator To Computer Connection
Before starting SourceGate II, you must connect a serial cable from a serial port of the computer to the RS-232 port of the emulator. This should be a "straight-thru" or "modem" type cable.

Emulator To Target System Connection
You do not have to have the emulator connected to a target system for stand-alone operation. However, if you wish to use your target system, you must connect the emulation pod to your target system.

Starting SourceGate II From Program Manager
To start SourceGate II from Program Manager, double-click on the SourceGate II icon located in the HMI program group. **This is the recommended method of starting SourceGate II.** You can also choose "File/Run" from the Program Manager menu bar and enter the path of the SG2.EXE file, along with any command line options, and then choose OK.

Starting SourceGate II From File Manager

To start SourceGate II from File Manager, go to the directory where the SourceGate II files are located, highlight the SG2.EXE file, and choose OK.

Command Line Options

The following command line options can be entered for the SourceGate II program. To add these parameters for the SourceGate II icon, highlight the SourceGate II icon by clicking on it once. Then choose 'File/Properties' and entered the desired command line options on the command line after the SG2.EXE filename. Be sure and leave a space between each option. After you are done adding options, choose OK to save the new properties. The next time SourceGate II is started by double-clicking on the SourceGate II icon, the new options will be in effect.

The available command line options are as follows:

-<Baud>
Select communications port baud rate. Allowable selections are: N(No serial port initialization performed), 110, 150, 300, 600, 1200, 2400, 4800, 9600 (default),19200, 38400, and 57600. **Note: Windows does not support baud rates higher than 57600 baud.**

-<Port>
Select communications port. SourceGate II automatically uses COM1 as its default communications port, unless COM1 is not available, in which case it will use COM2. The -port option will override the default SourceGate II port selection and cause SourceGate II to use only the port named. Allowable options are: COM1, COM2, COM3, COM4 and AUX.

-D<DownLoadPort>
Select parallel port for downloading. SourceGate II will default to PRN. Available options are LPT1, LPT2, and LPT3.

-L<LineSymbol>
In order to enable source-level debugging, SourceGate II must know the relationship between the lines in the source file on disk, and the assembly language code in the emulator. This relationship is defined through the use of Line Number Symbols.

The default format for the line number symbols recognized by SourceGate II is "L_%u". This is a scanf-formatted string indicating that line number symbols are those symbols which consist of one capital 'L' character, followed by an underscore character and an unsigned decimal number. The unsigned decimal number is the line number. The value associated with the symbol is the starting address of the assembly-language instruction generated by that line of source code.

If, for example, your compiler produced line number symbols of the format LL_%u, then the-LLL_%u option should be used on the command line which initiates SourceGate II. In this example, the -L tells SourceGate II that the line number format is LL_%u. This command line switch allows source-level debugging using any compiler that produces line number symbols.

-E<SrcExt>
This switch defines the source code extent. The default extent is ".C". When source level or mixed mode debugging is enabled, SourceGate II will look for source modules with the defined extent.

-[NO]RTS
This option enables/disables the RTS/CTS handshaking between the emulator and the computer system. If "-RTS" is used, the software will enable the RTS/CTS handshake. If "-NORTS" is used, the software will disable the RTS/CTS handshake. This hardware handshake may be needed if "COM PORT OVERRUN" messages appear at any time in a status window while transferring data.

-B<CommandFile>
This option defines a command (.CMD) file to be automatically executed upon entering SourceGate II.

<HexFile>
This option establishes the default program filename for RD and W commands.

Examples:

SG2.EXE	= Start SourceGate II with default options
SG2.EXE -19200	= Set 19200 Baud
SG2.EXE -COM2	= Use COM2
SG2.EXE -DLPT2	= Use LPT2 for parallel downloads
SG2.EXE -LLL_%u	= Set Line Number Symbol format to "LL_%u"
SG2.EXE -E68K	= Set extent default to ".68K"
SG2.EXE -NORTS	= Disable RTS/CTS handshake
SG2.EXE -BTEST	= Define "TEST.CMD" as command file to be run when SourceGate II is invoked
SG2.EXE DEMO.OUT	= Establish "DEMO.OUT" as default filename for RD and W commands

If all of the above options were used, the command line would look like the following:

SG2.EXE -19200 -COM2 -DLPT2 -LLL_%u -E68K -NORTS -BTEST DEMO.OUT

Environment Variables

Several environment variables are used by SourceGate II. These environment variables can be defined by using the DOS "SET" command. These variables are defined as follows:

HMI_BAUD:
Environment variable for the baud rate of the serial link between the computer and the emulator. Options are N(None), 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, and 57600. **Note: Windows does not support baud rates higher than 57600 baud.**

HMI_LINE:
Environment variable for line number format.

HMI_PORT:
Environment variable for communications port being used to communicate between the computer and the emulator. Options are COM1, COM2, COM3, COM4, and AUX.

HMI_LPT:
Environment variable for parallel download port to be used when downloading files from the computer to the emulator over the parallel link. Options are PRN, LPT1, LPT2, and LPT3.

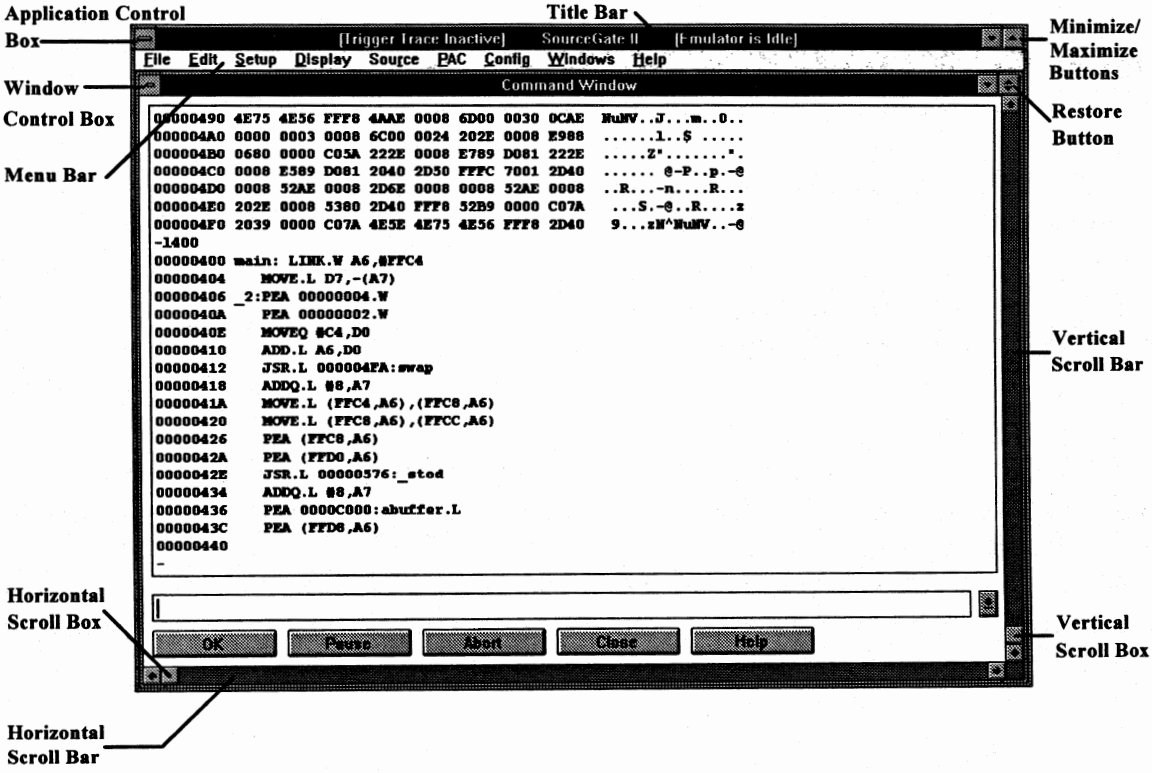
HMI_RTS:
Environment variable for default setting of the RTS/CTS handshaking between the computer and the emulator. Options are Y (enable handshake) and N (disable handshake).

HMI_STARTUP:
Environment variable for setting the command file that gets executed when SourceGate II is invoked. This is handy for automating the loading of a file when SourceGate II is run.

2 General Overview

Becoming Familiar With The SourceGate II Window

Take a moment to review the SourceGate II window on this page. Become familiar with the various parts of the SourceGate II window so you can work quickly and efficiently.



An explanation of each of these window controls is given as follows:

Control Menu Boxes

Displays a menu with commands that allow you to restore, move, size, minimize, maximize, or close the application/active window. The application control menu box also allows you to switch between windows or open the control panel window.

Menu Bar

Lists the main menus in SourceGate II.

Title Bar
Displays the name of the application (SourceGate II) and shows the status of the emulator (Idle or Running) and the trace trigger (Active or Inactive).

Restore Button
Restores the SourceGate II application or window to the size and position it occupied before you maximized or minimized it.

Maximize Button
Enlarges the SourceGate II application or window to fill the entire screen.

Minimize Button
Reduces the SourceGate II application or window to an icon at the bottom of the screen.

Scroll Bars (Horizontal and Vertical)
Controls the portion of window text that appears on the screen.

Scroll Boxes (Horizontal and Vertical)
Indicates where the screen display is located compared to the full width of the window.

SourceGate II Menu Bar
The menu bar for SourceGate II provides you a way to pull up the main menus that can be used with SourceGate II. An explanation of each of these menus follows. Note that the menu bar choices might change depending on the current active window. The following menu choices are available when the main SourceGate II window is active. **If a menu item is greyed out, it is either not available on this particular emulator or the menu item has not been implemented in this version of SourceGate II.**

The 'File' Menu
The main function of the 'File' menu is getting information into and out of SourceGate II. The following is a brief explanation of the options available in this menu:

Download Hex File
Download an object file into memory.

Upload Hex File
Upload memory into an object file that can be used to burn PROMs.

Execute
Execute a pre-recorded command file.

Record
Record a command file.

Log
Begin logging all keyboard input and screen output to an ascii file that can be reviewed at a later date. This enables you to obtain a "log" file of what you have done.

Print
Print the contents of the active window to a file or a printer. **This menu item will be greyed out if the active window does not support printing operations.**

Printer Setup
Setup printer parameters.

Exit SourceGate
Exit the SourceGate II program.

The 'Edit' Menu
The 'Edit' menu will allow you to cut and paste information from/to the SourceGate II windows. This menu is not implemented in this version of SourceGate II.

The 'Setup' Menu
The 'Setup' menu is used to "setup" the event, breakpoint, and trace trigger windows. The following is a brief explanation of the options available in this menu:

Event
Define the conditions of the Event windows A, B, C, and D.

Breakpoint
Define the condition to break emulation.

Trace Trigger
Define the condition to trigger the trace.

The 'Display' Menu
The 'Display' menu allows the user to choose certain windows for display. The following is a brief explanation of the options available in this menu:

Command Window
Open the Command Window and make it the active window.

Trace
Open the Trace Window and make it the active window.

Watch
Open the Watch Window and make it the active window.

Watch In Real Time
Open the Watch In Real Time Window and make it the active window.

Registers
Open the Register Window and make it the active window.

Memory
Open the Memory Window and make it the active window.

Stack Trace
Open the Stack Trace Window and make it the active window.

Run Status
Open the Run Status Window and make it the active window.

The 'Source' Menu

The main function of the 'Source' menu is to provide an interface when dealing with source code information. The following is a brief explanation of the options available in this menu:

Run
Begin emulation from the current program counter location. This item will be greyed out if emulation is in progress.

Go From/To
Open the 'Go From/To' window which allows the user to begin emulation from a specified address and also allows temporary breakpoints to be set.

Step
Open the 'Single Step' window which allows the user to single step through instructions.

Stop
Stop emulation and update all window values with the latest information. This item will be greyed out if emulation is not in progress.

Reset Communications
Reset the communications between the emulator and SourceGate II. This may be necessary if the emulator and SourceGate II were to get out of "sync" for some reason.

Reset Emulator
Reset the emulator's processor by passing the RESET signal to both the processor and the target system.

Symbols
Open the 'Symbol Selection' window which allows you to view and select symbols for a particular module.

Modules
Open the 'Module Selection' window which allows you to view and select a source module to work with.

Assemble
Open an assembly window for the current module. You can then work at the assembly level.

VASM
Open an assembly window beginning at the address defined by the user.

Tracking Assembly
Invoke an assembly window which will remain on the screen and will track the source window that currently has the focus by highlighting the assembly lines that correspond to the source line highlighted in the active module. If this window is activated when no source modules are open, it will show assembly code beginning at the current program counter location.

The 'Config' Menu
The 'Config' menu deals with setting up emulator options. The following is a brief explanation of the options available in this menu:

Memory
Configure the emulator's memory map and other optional signal parameters.

Source Code Interface
Define the relationship between SourceGate II and the compiler being used.

The 'Windows' Menu

The 'Window' menu allows you to manipulate the SourceGate II windows. The following is a brief explanation of the options available in this menu:

Next Window
Open the next window in the open window list of previously opened windows.

Previous Window
Make the window that was active prior to the current window to be the active window.

Cascade Windows
Cascades all open windows.

Close
Close the open window that currently has focus.

Close All
Close all open windows.

Open Window List
Not an actual option, this list will show all windows that are currently open. You can make a window in this list to be the active window by clicking on it. If no windows are open, "No Windows" will be shown.

The 'Help' Menu

The 'Help' menu provides help on using SourceGate II. The following is a brief explanation of the options available in this menu:

On Context
Provides help on the current active window. This is the same as 'On Window'. Future versions of SourceGate II may use this option to provide help on particular fields instead of at the window level.

On Window
Provides help on the current active window. This is the same as 'On Context'.

Keyboard
Provides help on commands that can be executed from the keyboard.

Index
Shows an index of the topics available in the SourceGate II Help System.

How Do I...
Provides help for some of the more commonly asked questions.

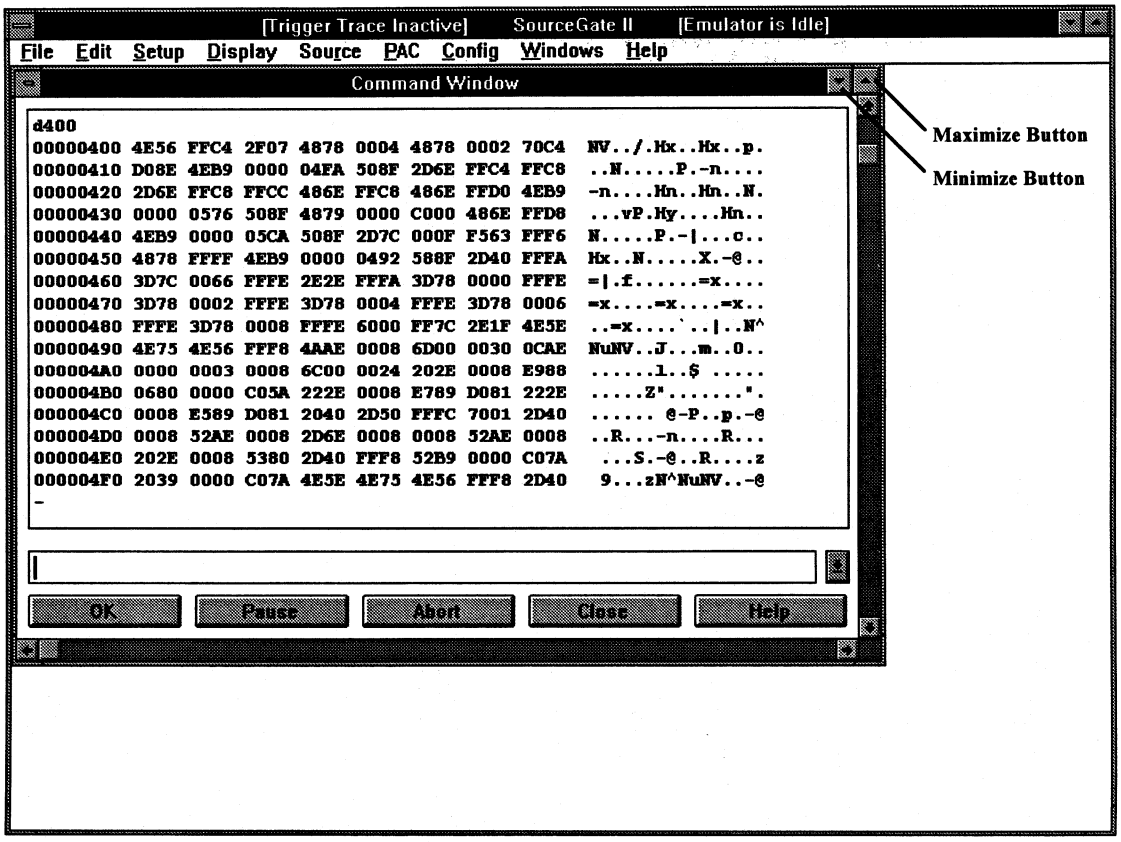
Help On Help
Provides help on how to use the help system.

Search
Allows you to search for a particular topic by using keywords.

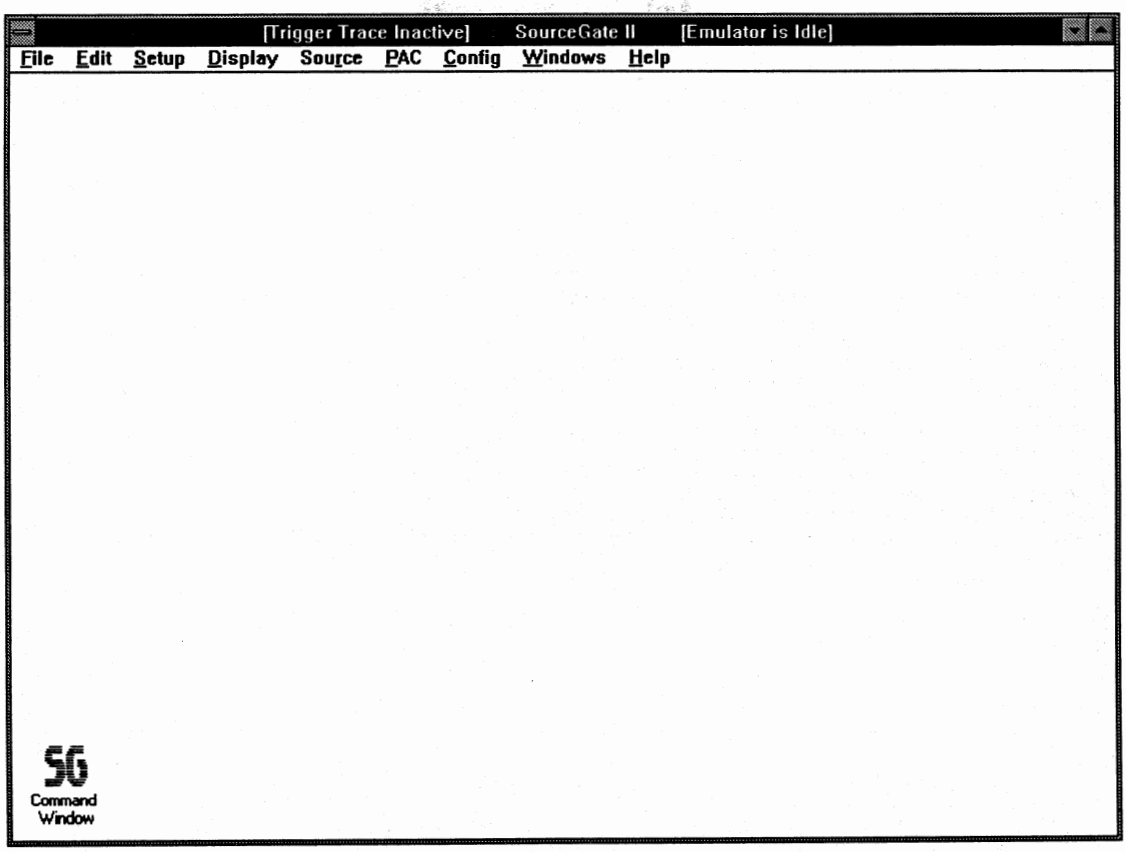
About
Shows information about the version of the help files.

Maximizing/Minimizing Windows

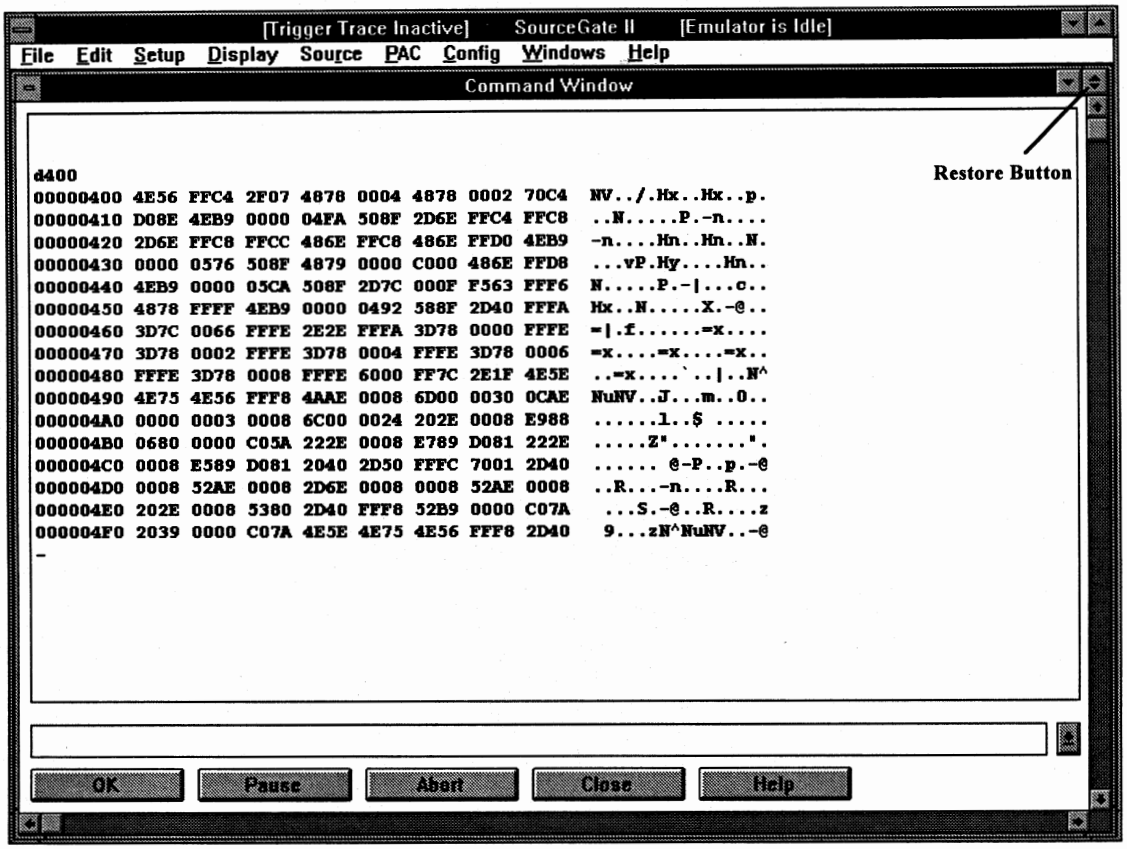
By using the maximize and minimize buttons, you can make SourceGate II windows to be icons or windows of a certain size. An example may help. Consider the main SourceGate II window shown below:



Notice that the Command window has a minimize and a maximize button. If the minimize button is pressed, the Command window will be reduced to an icon as shown in the following example:



You can double-click on the icon to make the window active again. If the maximize button is then pressed, the Command window will take on its largest size as shown in the following example:



Pressing the restore button will restore the Command window back to its original size.

Windows can also be custom sized by clicking and dragging the window. For more information on how to do this, consult the Windows manual.

Window Manipulation

Several windows can be shown at the same time. However, you may wish one window to "have focus" or be the top-most window. The following sections describe how to work with multiple windows.

Changing To The Next Window

Once several windows have been opened, you may choose the 'Windows' menu command. You will then be shown a listing of the windows that are open. A check mark will be next to the active window. To make the window listed below the checked one to be the active window, choose ***Next Window***.

Changing To The Previous Window
Once several windows have been opened, you may choose the 'Windows' menu command. You will then be shown a listing of the windows that are open. A check mark will be next to the active window. To make the window listed above the checked one to be the active window, choose ***Previous Window.***

Choosing From The Open Window List
Once several windows have been opened, you may choose the 'Windows' menu command. You will then be shown a listing of the windows that are open. A check mark will be next to the active window. To make another window in the list to be the active window, simply click on it.

Cascading All Open Windows
To cascade all open windows, choose ***Cascade Windows.***

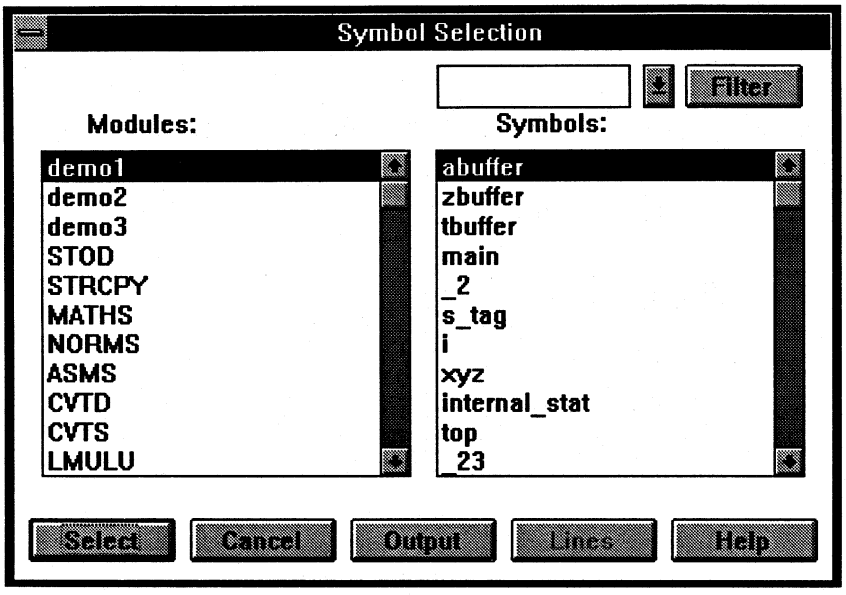
Closing The Current Active Window
To close the current active window, choose ***Close.***

Closing All Open Windows
To close all windows that are currently open, choose ***Close All.***

Defining Your Printer
You can define the printer you wish to use with SourceGate II by choosing 'File/Printer Setup'. You will be shown the current printer and you can then change to another printer, if you wish, provided you have defined an alternate printer using the Windows 'Setup' group. For additional information on how to do this, see the Windows manual.

Working With Symbols

While you are in any field in SourceGate II from which symbols can be entered, pressing the ***right mouse button*** opens the Symbol Selection window which allows you to select a symbol for that field. An example Symbol Selection window is shown below:



From this window, you can select one of the pre-defined symbols by clicking on it and then pressing the **Select** button. The selected symbol will then appear in the field from which this window was invoked.

Using Help

Using the Microsoft Windows Help Program, SourceGate II provides online Help for every window. You can resize, move, tile, or cascade the Help window and the current active window so that you can keep both of them displayed.

There are several ways to obtain online Help:

- Context sensitive Help
- Help for the current active window
- Help menu

Currently, 'context sensitive help' and 'help for the current active window' perform the same function. Help is always provided for the current active window if either of these options is chosen. You can always use the accelerator key F1 to obtain help on the current active window.

Context Sensitive Help

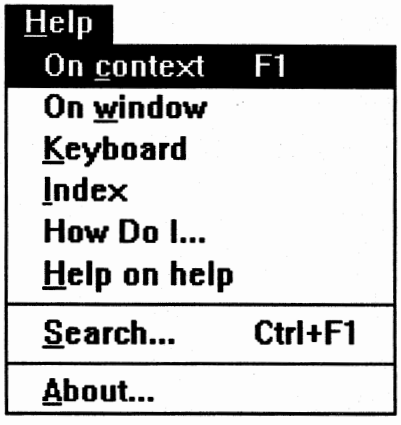
Choose 'Help/On Context'. SourceGate II will then provide Help for the active window (see ***Active Window Help***).

Active Window Help

Choose 'Help/On Window'. SourceGate II will then provide Help for the active window.

Help Menu

The Help menu will provide help on all areas. To invoke the Help menu, choose 'Help'. The following menu will then be displayed:



The following is a description of each available menu item:

On Context

Provides help on the current active window (same as ***On Window*** for this release).

On Window

Provides help for the current active window.

Keyboard

Provides keyboard shortcuts (accelerator keys) for SourceGate II functions and commands.

Index

Displays a list of available help topics from which you can gain access to other parts of the SourceGate II Help System.

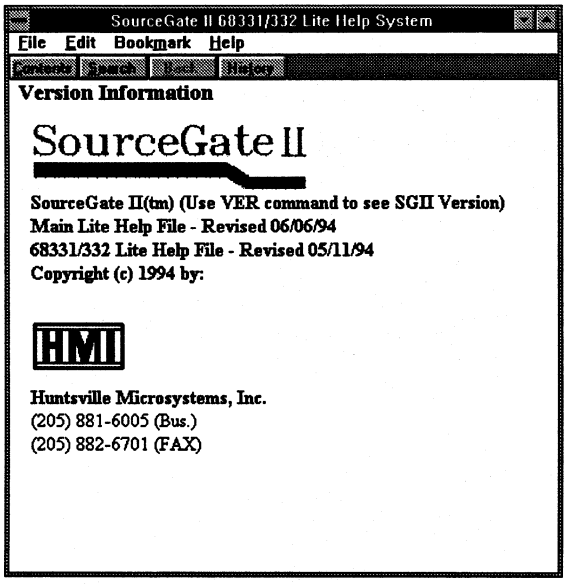
How Do I...

Displays a list of common SourceGate II questions and provides answers to each.

Help On Help
Switches control from SourceGate II Help to Windows Help and displays information about using Windows Help.

Search
Invokes the Help system's Search engine allowing you to search for a particular topic or keyword.

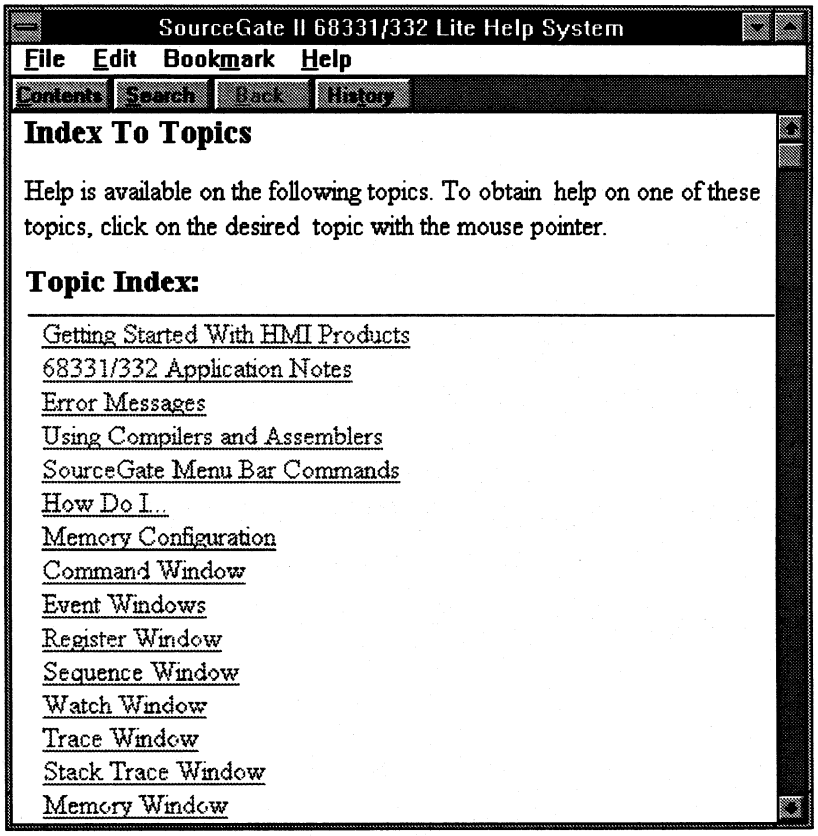
About
Provides information about the version of the Help files. The following is an example of what you may see when you choose this menu item:


SourceGate II 68331/332 Lite Help System
File Edit Bookmark Help
Version Information
SourceGate II
SourceGate II(tm) (Use VER command to see SGII Version)
Main Lite Help File - Revised 06/06/94
68331/332 Lite Help File - Revised 05/11/94
Copyright (c) 1994 by:
HMI
Huntsville Microsystems, Inc.
(205) 881-6005 (Bus.)
(205) 882-6701 (FAX)

Notice that the version of SourceGate II is not shown in this window. To find out what version of SourceGate II that you are using, use the "VER" command.

SourceGate II Help Window

An example SourceGate II Help window is shown as follows:


SourceGate II 68331/332 Lite Help System
File Edit Bookmark Help
Contents Search Back History
Index To Topics
Help is available on the following topics. To obtain help on one of these topics, click on the desired topic with the mouse pointer.
Topic Index:
Getting Started With HMI Products
68331/332 Application Notes
Error Messages
Using Compilers and Assemblers
SourceGate Menu Bar Commands
How Do I...
Memory Configuration
Command Window
Event Windows
Register Window
Sequence Window
Watch Window
Trace Window
Stack Trace Window
Memory Window

An explanation of the different parts of this window follows:

Help Window Buttons

You can use the Help buttons to display related Help topics. Each button is described as follows:

Contents

Displays a list of main SourceGate II Help topics from which you can gain access to other parts of the SourceGate II Help System.

Search

Lists all of the Help keywords for SourceGate II . Enter a keyword or phrase in the 'Search For' text box or select a keyword or phrase from the list box.

Back
Displays the last Help topic you viewed. You can move back through each topic, one at a time. When you are at the first topic you viewed, this button is dimmed.

History
Displays a list of the previous Help topics you viewed. You can choose any topic in the list to display Help for that topic.

Help Window Menus
When the Help window is active, the Help menu bar displays the available menus. Each of these menu choices is defined as follows:

File
Opens Help files, prints Help topics, sets printer options, and closes Help.

Edit
Copies Help text to the Clipboard and adds annotations to Help text.

Bookmark
Places bookmarks in and removes bookmarks from Help topics. You use a bookmark to mark a specific Help reference.

Help
Switches control from SourceGate II Help to Windows Help and displays information about using Windows Help. To return to SourceGate II Help, choose **History** and then choose the desired SourceGate II Help topic.

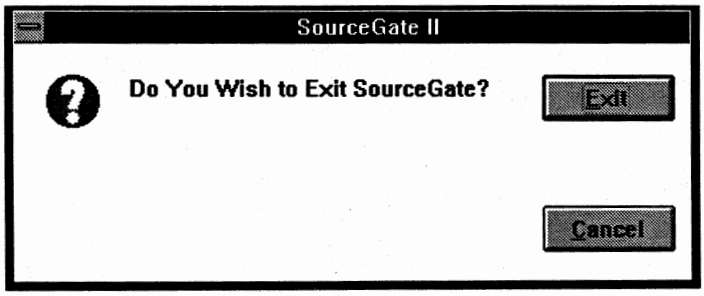
Cross-References
Many Help topics contain cross-references to other related Help topics. The cross-references display in green (on a color monitor) with a solid underline. To choose a cross-reference, simply click on it. SourceGate II will then display the related Help topic.

Exiting SourceGate II

To exit SourceGate II , you can do one of the following:

- Choose 'File/Exit SourceGate'
- Press ALT+F4
- Double-Click on the SourceGate II window application control box

After performing one of the above steps, you will be asked if you really want to exit SourceGate II :

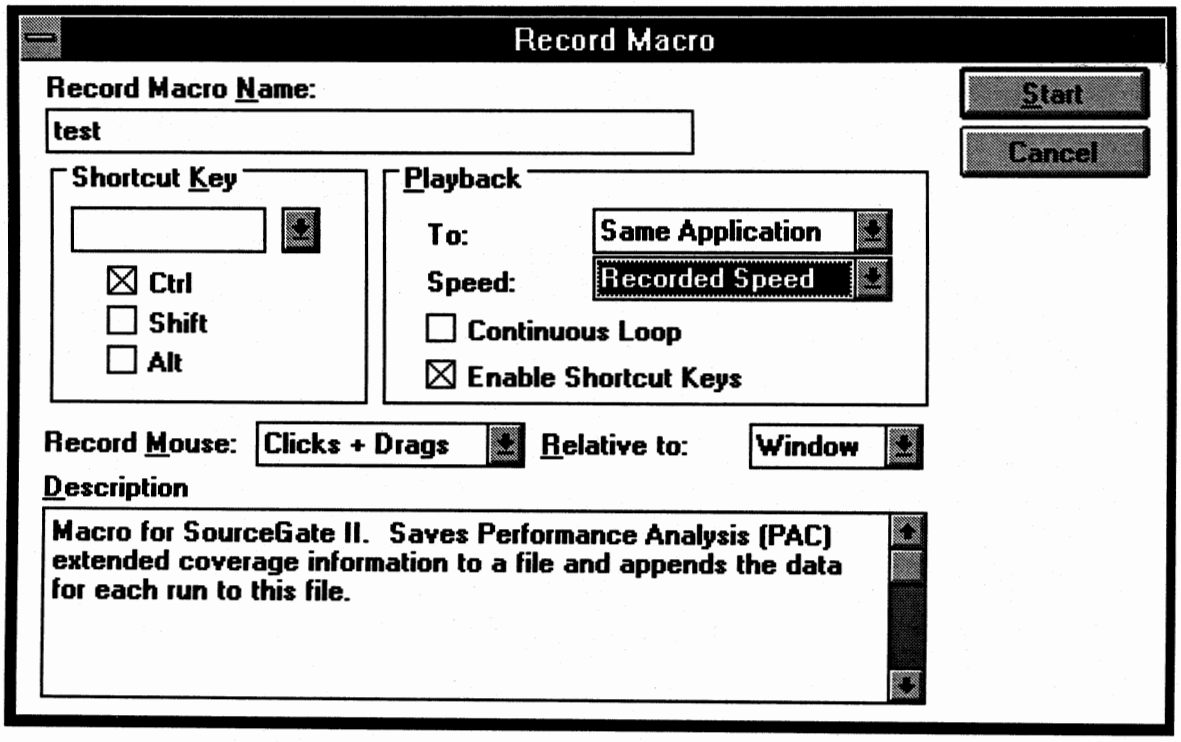


Choosing 'Exit' will exit SourceGate II . All active window sizes and positions will be saved so that they will appear the same the next time SourceGate II is started.

Automating SourceGate II

Using the Windows Recorder, SourceGate II operations can be automated. This will allow you to perform a series of commands, such as opening windows, performing commands in these windows, saving information to a file, etc. This could prove to be invaluable when setting up automatic testing programs.

The following options in the Windows Recorder "Record Macro" window have been shown to work well when recording SourceGate II operations. It is very important that you use the "Recorded Speed" option in order for the playbacks to function correctly. For additional information on using the Windows Recorder, see your Microsoft® Windows manual.



Accelerator Keys

You can use accelerator keys, shown beside some commands, to select a command. For example, ALT+F4 will exit SourceGate II. The following is a listing of all available accelerator key combinations.

This key...	**...does this**
Alt+D	Change Trace Format To "Disassembly"
Alt+O	Change Trace Format To "Source"
Alt+S	Change Trace Format To "State"
Alt+F4	Exit SourceGate II
Ctrl+A	Activate Assembly Window
Ctrl+B	Activate Breakpoint Window
Ctrl+C	Activate Command Window
Ctrl+D	Perform 'File/Download Hex File' Function
Ctrl+E	Activate Event A Window
Ctrl+F	Perform 'File/Execute' Function
Ctrl+G	Activate Trace Trigger Window
Ctrl+H	Peform 'Source/Stop' Function
Ctrl+J	Perform 'File/Record' Function
Ctrl+K	Activate Stack Trace Window
Ctrl+L	Perform 'File/Log' Function
Ctrl+M	Activate Memory Window
Ctrl+N	Activate Tracking Assembly Window
Ctrl+O	Activate Module Selection Window
Ctrl+Q	Perform 'Setup/Event/Clear All' Function
Ctrl+R	Activate Register Window
Ctrl+S	Activate Run Status Window
Ctrl+T	Activate Trace Window
Ctrl+U	Perform 'File/Upload Hex File' Function
Ctrl+V	Activate VASM Window
Ctrl+W	Activate Watch Window
Ctrl+Y	Activate Symbol Selection Window
Ctrl+Z	Perform 'Windows/Close All' Function
Ctrl+F1	Perform 'Help/Search' Function
Shift+F6	Activate Previous Window
F6	Activate Next Window
F1	Get Help On Active Window
F5	Perform 'Source/Run' Function From Current PC

3 The Emulation Interface

SourceGate II and the emulator it is controlling have a close relationship. When you enter a command in SourceGate II, it must be translated into a command that the emulator can understand. Likewise, when the emulator sends information to the computer, SourceGate II must take this information and display it to you. This partnership makes up what we refer to as the "Emulation Interface". In this chapter, we will attempt to explain the relationship between the emulator and SourceGate II, and we will show you how to set up SourceGate II to perform the emulation tasks you require. Each area of the "Emulation Interface" will be explained in the sections that follow.

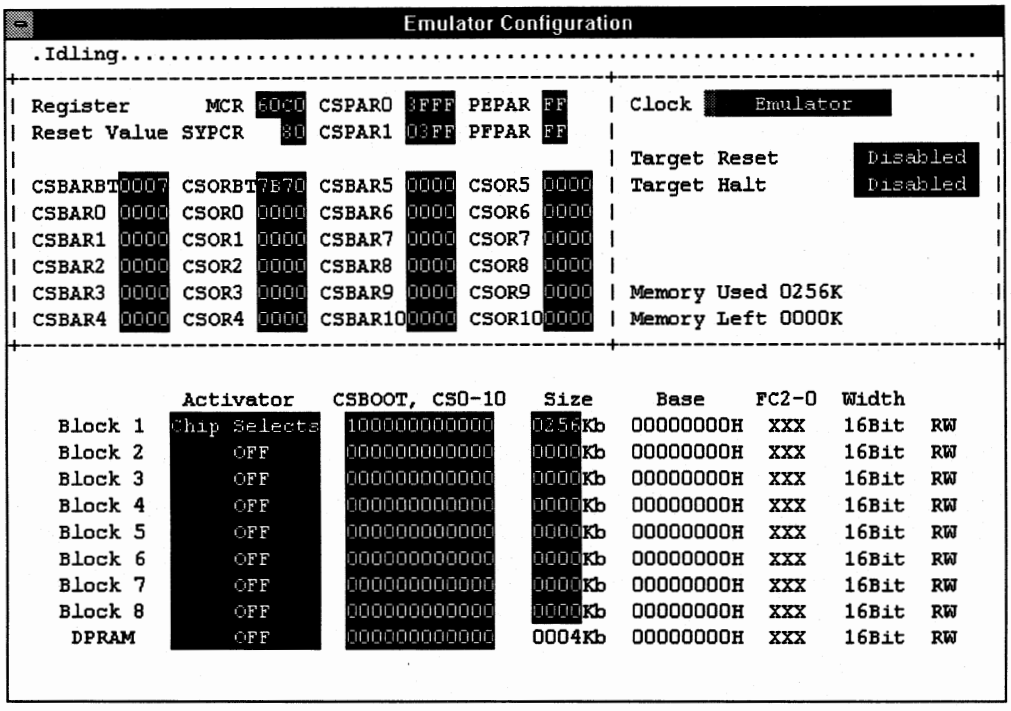
Defining The Emulator's Configuration Window

Each emulator has a Configuration window that allows you to set up the memory map and define how certain signals are handled. To open this window, choose 'Config/Memory'. There are several Configuration windows that may be shown depending on the version of the *Lite* emulator being used. An example of each of these is shown in the following sections.

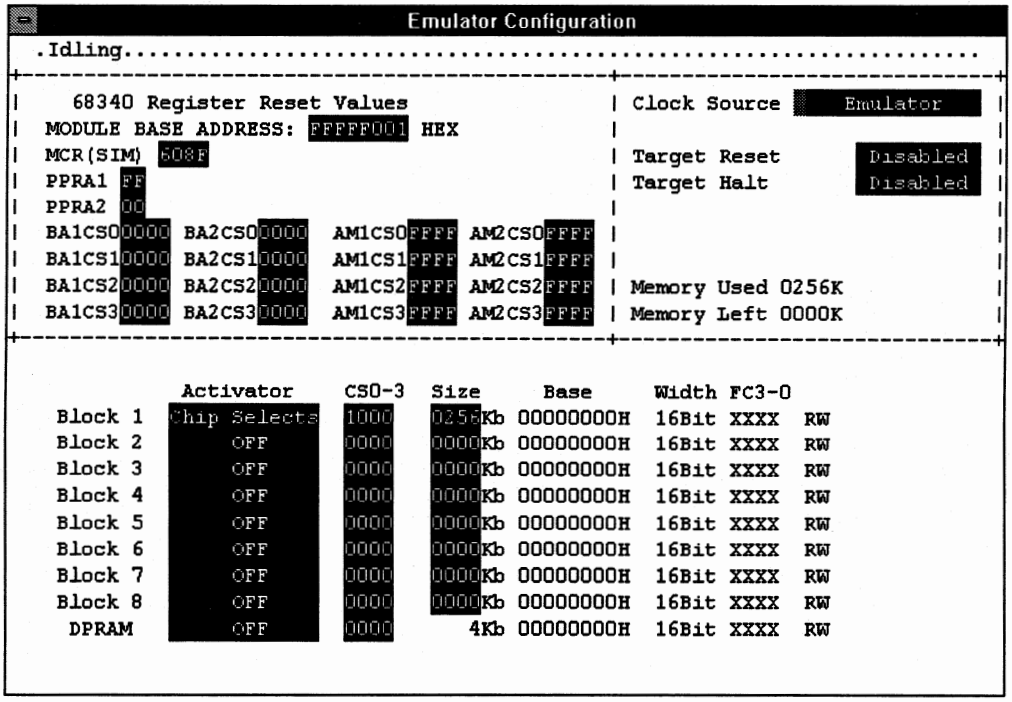
Terminal Type Configuration Windows

Some *Lite* emulators being used with SourceGate II will show a terminal type Configuration window. The information shown in this window will vary depending on what version of the *Lite* emulator that you are using. The following pages will show Configuration windows for each *Lite* emulator version.

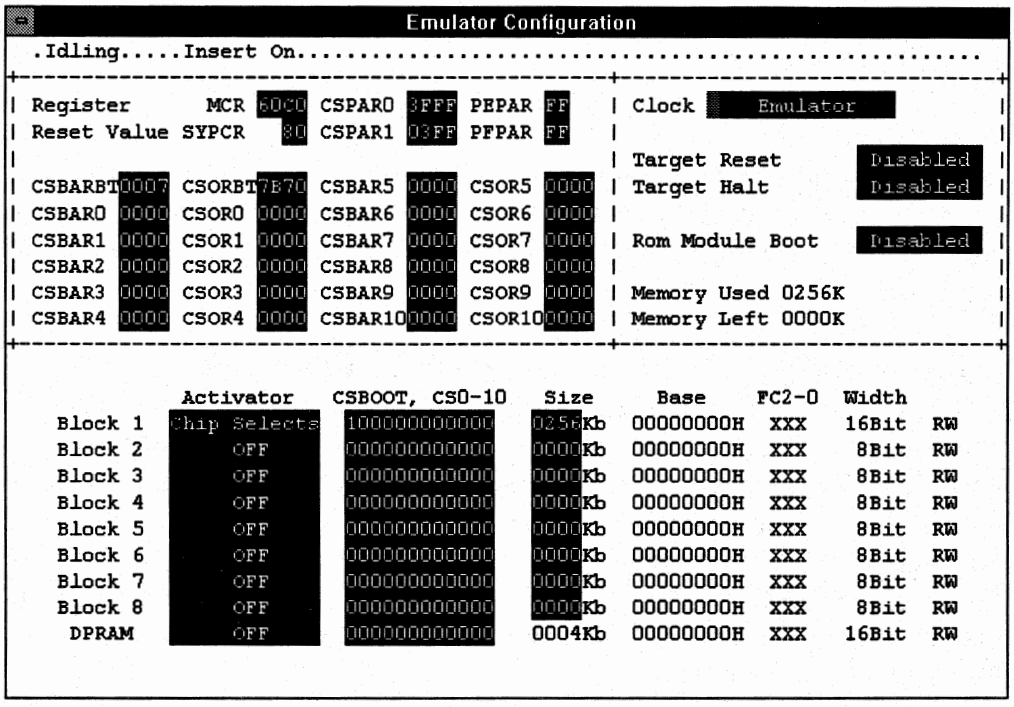
If you are using a 68331 or 68332 *Lite* emulator, the following Configuration window will be shown:


Emulator Configuration
.Idling.....
Register MCR 0C0 CSPAR0 3FFF PEPAR FF
Reset Value SYPCR 80 CSPAR1 03FF PFPAR FF
CSBARBT 0007 CSORBT 7B70 CSBAR5 0000 CSOR5 0000
CSBAR0 0000 CSOR0 0000 CSBAR6 0000 CSOR6 0000
CSBAR1 0000 CSOR1 0000 CSBAR7 0000 CSOR7 0000
CSBAR2 0000 CSOR2 0000 CSBAR8 0000 CSOR8 0000
CSBAR3 0000 CSOR3 0000 CSBAR9 0000 CSOR9 0000
CSBAR4 0000 CSOR4 0000 CSBAR10 0000 CSOR10 0000
Clock Emulator
Target Reset Disabled
Target Halt Disabled
Memory Used 0256K
Memory Left 0000K
Activator CSBOOT, CS0-10 Size Base FC2-0 Width
Block 1 Chip Selects 10000000000 0256Kb 00000000H XXX 16Bit RW
Block 2 OFF 00000000000 0000Kb 00000000H XXX 16Bit RW
Block 3 OFF 00000000000 0000Kb 00000000H XXX 16Bit RW
Block 4 OFF 00000000000 0000Kb 00000000H XXX 16Bit RW
Block 5 OFF 00000000000 0000Kb 00000000H XXX 16Bit RW
Block 6 OFF 00000000000 0000Kb 00000000H XXX 16Bit RW
Block 7 OFF 00000000000 0000Kb 00000000H XXX 16Bit RW
Block 8 OFF 00000000000 0000Kb 00000000H XXX 16Bit RW
DPRAM OFF 00000000000 0004Kb 00000000H XXX 16Bit RW

If you are using a 68340 *Lite* emulator, the following Configuration window will be shown:


Emulator Configuration
.Idling.....
68340 Register Reset Values
MODULE BASE ADDRESS: FFFFF001 HEX
MCR(SIM) 608F
PPRA1 FF
PPRA2 00
BA1CS0 0000 BA2CS0 0000 AM1CS0 FFFF AM2CS0 FFFF
BA1CS1 0000 BA2CS1 0000 AM1CS1 FFFF AM2CS1 FFFF
BA1CS2 0000 BA2CS2 0000 AM1CS2 FFFF AM2CS2 FFFF
BA1CS3 0000 BA2CS3 0000 AM1CS3 FFFF AM2CS3 FFFF
Clock Source Emulator
Target Reset Disabled
Target Halt Disabled
Memory Used 0256K
Memory Left 0000K
Activator CS0-3 Size Base Width FC3-0
Block 1 Chip Selects 1000 0256Kb 00000000H 16Bit XXXX RW
Block 2 OFF 0000 0000Kb 00000000H 16Bit XXXX RW
Block 3 OFF 0000 0000Kb 00000000H 16Bit XXXX RW
Block 4 OFF 0000 0000Kb 00000000H 16Bit XXXX RW
Block 5 OFF 0000 0000Kb 00000000H 16Bit XXXX RW
Block 6 OFF 0000 0000Kb 00000000H 16Bit XXXX RW
Block 7 OFF 0000 0000Kb 00000000H 16Bit XXXX RW
Block 8 OFF 0000 0000Kb 00000000H 16Bit XXXX RW
DPRAM OFF 0000 4Kb 00000000H 16Bit XXXX RW

If you are using a 68HC16Z1 or 68HC16Z2 *Lite* emulator, the following Configuration window will be shown:


Emulator Configuration
.Idling.....Insert On.....
Register MCR 60C0 CSPAR0 3FFF PEPAR FF
Reset Value SYPCR 80 CSPAR1 03FF PFPAR FF
CSBARBT 0007 CSORBT 7B70 CSBAR5 0000 CSOR5 0000
CSBAR0 0000 CSOR0 0000 CSBAR6 0000 CSOR6 0000
CSBAR1 0000 CSOR1 0000 CSBAR7 0000 CSOR7 0000
CSBAR2 0000 CSOR2 0000 CSBAR8 0000 CSOR8 0000
CSBAR3 0000 CSOR3 0000 CSBAR9 0000 CSOR9 0000
CSBAR4 0000 CSOR4 0000 CSBAR10 0000 CSOR10 0000
Clock Emulator
Target Reset Disabled
Target Halt Disabled
Rom Module Boot Disabled
Memory Used 0256K
Memory Left 0000K
Activator CSBOOT, CS0-10 Size Base FC2-0 Width
Block 1 Chip Selects 10000000000 0256Kb 00000000H XXX 16Bit RW
Block 2 OFF 00000000000 0000Kb 00000000H XXX 8Bit RW
Block 3 OFF 00000000000 0000Kb 00000000H XXX 8Bit RW
Block 4 OFF 00000000000 0000Kb 00000000H XXX 8Bit RW
Block 5 OFF 00000000000 0000Kb 00000000H XXX 8Bit RW
Block 6 OFF 00000000000 0000Kb 00000000H XXX 8Bit RW
Block 7 OFF 00000000000 0000Kb 00000000H XXX 8Bit RW
Block 8 OFF 00000000000 0000Kb 00000000H XXX 8Bit RW
DPRAM OFF 00000000000 0004Kb 00000000H XXX 16Bit RW

Each of the fields shown in the preceeding windows are outlined in the sections that follow.

Terminal Type Configuration Window Fields

Many of the fields shown in these windows must be configured differently to work with specific target systems. Any field that is highlighted can be changed by moving the cursor to that field and pressing the space bar. Each field in these windows is described in the sections that follow:

Clock Field

The source of the clock can be the "Emulator" on-board clock or the "Target System" clock.

Target Reset Field

This field allows the RESET line from the target system to reset the microprocessor in the *Lite* if "Enabled" is selected. If "Disabled" is selected, the *Lite* will not allow the target system's reset line to reset the microprocessor.

Target Halt Field

This field allows the HALT line from the target system to influence the microprocessor in the *Lite* if "Enabled" is selected. If "Disabled" is selected, the *Lite* will not allow the target system's halt line to influence the microprocessor.

Rom Module Boot Field (68HC16Z1/Z2 Only)

68HC16Z1 and 68HC16Z2 *Lite* units have this field that, when "Enabled", will activate the on-chip Boot ROM. When this field is defined to be "Disabled", the on-chip Boot ROM is deactivated and external memory can be configured to be the Boot ROM. **Note that this field can be enabled for the 68HC16Z1 processor, but it performs no fuction since this processor contains no on-chip Boot ROM.**

Register Reset Values Fields

The values normally programmed into the SIM registers should be defined here. Any time the user performs a reset of the processor, the SIM registers will be programmed with the values entered here. These registers should contain the same information as the user's startup code to avoid conflicts.

Terminal Type Configuration Window Memory Map Fields

The bottom portion of the Configuration menu allows the user to define up to 8 blocks in addition to a dual-port RAM (DPRAM) block that works in conjunction with the Watch In Real Time window. These blocks can be configured to define blocks used as actual memory or for the definitions of chip selects. When any of these blocks are modified, the emulator immediately issues a reset to the processor in order to program the new conditions and the message "PROGRAMMING" is displayed. If there are conflicts between what is defined here and what is programmed in the SIM registers, the message "Possible invalid selection(s) in Config. Menu." will be displayed.

NOTE: The values programmed for the SIM registers should reflect the values being placed in these registers by the user's startup code. Also, the user must set up these registers BEFORE downloading any files. If these registers are not defined first, the chip selects will not be mapped correctly and the code will not load properly.

Activator Fields

The Activators allow you to define the conditions which enable emulation memory. There are 8 blocks that can be defined in addition to a dual-port RAM (DPRAM) block that works in conjunction with the Dual-Port Watch window. The "Activator" field determines what combination of parameters that must occur before the defined block of memory will be activated. Each of these activators is described as follows:

68331/68332 and 68HC16Z1/Z2 Activators:

The Activator field is used to select between "Chip Selects", "CS, Base, FC", and "OFF". This selection is accomplished by pressing the space bar to toggle through the available choices. If "Chip Selects" is chosen as the activator, the associated block of emulation memory will become active when the chip- select defined with a "1" in the "CSBOOT, CS0-10" field is active. If "CS, Base, FC" is chosen as the activator, the block will become active when a "AND" combination of the "CSBOOT, CS0-10", "Base", and "FC2-0" fields is valid. If "OFF" is selected as the activator, the block definition is ignored.

68340 Activators:
The Activator field is used to select between "Chip Selects", "CS and Base", and "OFF". This selection is accomplished by pressing the space bar to toggle through the available choices. If "Chip Selects" is chosen as the activator, the associated block of emulation memory will become active when the chip- select defined with a "1" in the "CS0-3" field is active. If "CS and Base" is chosen as the activator, the block will become active when a "AND" combination of the "CS0-3" and "Base" fields is valid. If "OFF" is selected as the activator, the block definition is ignored.

Dual-Port RAM (DPRAM) Activator:
The DPRAM activator is used to map the Lite's 4K of DPRAM. This block of memory can be mapped anywhere in the address space that you desire. This DPRAM is used with the Watch in Real Time window to "watch" certain variables during emulation.

CSBOOT, CS0-10 Field *(68331/332 and 68HC16Z1/Z2)*
If "Chip Selects" or "CS, Base, FC" is chosen in the activator field, the corresponding chip select signal should be enabled by placing a "1" in the column of the chip select that the user wishes to be used as the activator for the emulation memory block. CSBOOT is the left-most column followed by CS0-CS10 where CS10 is the right-most column.

This field can be influenced by the CSPAR0 and CSPAR1 registers. If a chip select is re-defined to its alternate function in one of these registers, the corresponding column for that chip select will be disabled and it cannot be used to enable an emulation memory block.

CS0-3 Field *(68340)*
If "Chip Selects" or "CS and Base" is chosen in the activator field, the corresponding chip select signal should be enabled by placing a "1" in the column of the chip select that the user wishes to be used as the activator for the emulation memory block. CS0' is the left-most column followed by CS1'-CS3' where CS3' is the right-most column.

This field can be influenced by the FIRQ bit in the MCR register. If chip selects are disabled by setting the FIRQ bit in the MCR register, the chip select fields will be blanked and they cannot be used to enable an emulation memory block.

Size Field
The "Size" field defines the size of the block to be defined. This field tells the Lite how much emulation memory is to be enabled when the specified activator is valid. It should be noted that the minimum block size allowed is 16K except for the DPRAM which is fixed at 4K. The space bar should be pressed to toggle through the available choices.

Base Field
If "Chip Selects" is defined as the activator, the "Base" field will show the current base address defined for the chip select enabled in the "CSBOOT, CS0-10" ("CS0-3" for the 68340) field. This should reflect the value programmed into the base address register for that chip select as defined in the Register Reset Values located in the upper left-hand corner of the Configuration window. It should be noted that the base field will always be on a 16K boundary since the minimum block size is 16K.

If "CS, Base, FC" ("CS and Base" for the 68340) is defined as the activator, you can enter the upper 5 digits of the base address.

Note: If a question mark (?) appears in a field, the combination selected may be invalid.

FC2-0 Field *(68331/332 and 68HC16Z1/Z2)*
If "Chip Selects" is defined as the activator, the "FC2-0" field will show the function code definition for the chip select enabled in the "CSBOOT, CS0-10" field. This should reflect the value programmed into the chip select option register for that chip select as defined in the Register Reset Values located in the upper left- hand corner of the Configuration window.

If "CS, Base, FC" is defined as the activator, you can enter the desired function code pattern.

Note: If a question mark (?) appears in a field, the combination selected may be invalid.

FC3-0 Field *(68340)*
If "Chip Selects" or "CS and Base" is defined as the activator, the "FC3-0" field will show the function code definition for the chip select enabled in the "CS0-3" field. This should reflect the value programmed into the base address and address mask registers for that chip select as defined in the Register Reset Values located in the upper left-hand corner of the Configuration window.

Note: If a question mark (?) appears in a field, the combination selected may be invalid.

Width Field
The "Width" field will show the width of the defined chip select. This should reflect the value programmed into the CSPAR0 and CSPAR1 registers (address mask registers in the 68340) in the Register Reset Values in the upper left-hand corner of the Configuration window. It is not user-changable.

RO and RW Fields
This field will show whether the defined chip select is read-only (RO), write-only (WO), or read- write (RW). This should reflect the value programmed into the chip select option register for that chip select as defined in the Register Reset Values located in the upper left-hand corner of the Configuration window.

Note: If a question mark (?) appears in a field, the combination selected may be invalid.

Memory Used and Memory Left Fields

The total memory allocated in defined emulator memory blocks may not exceed the total emulation memory. "Memory Used" and "Memory Left" is displayed in the middle of the menu for quick reference.

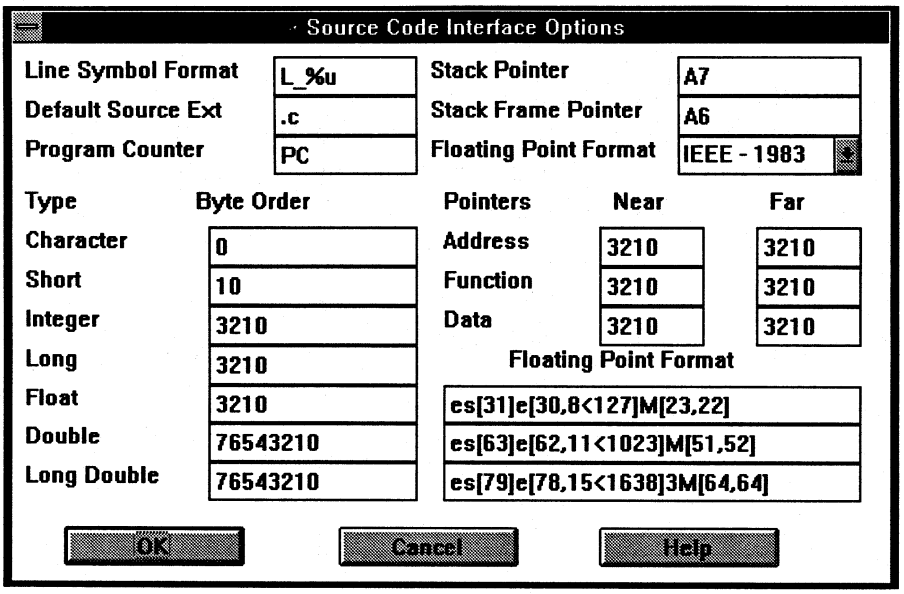
If the block size specified exceeds the amount of unused memory, the message "OUT OF MEMORY" will be displayed.

Graphical Configuration Windows

Future versions of this emulator may have graphical Configuration windows that contain the same information that was discussed in the previous sections. Currently, there are no graphical Configuration windows available for this emulator.

Source Code Interface Configuration

The Source Code Interface window is used to configure SourceGate II to work with almost any compiler. You can define several compiler specific parameters here such as the registers to be used for stack and frame pointers, the line symbol format used by the compiler, etc. To open this window, choose 'Config/Source Code Interface'.



Working With Object Files

The heart of any emulation system is the code that is loaded into memory. In the case of a working target system, this code usually will reside in a programmable device such as a PROM. However, when a system is under development, burning new PROMs every time the software changes can become a time-consuming chore. With SourceGate II and a HMI emulator, this unpleasant task is a thing of the past until you have the final working version of your code developed.

SourceGate II can read a variety of object file formats directly into memory without requiring a conversion process. The formats supported include:

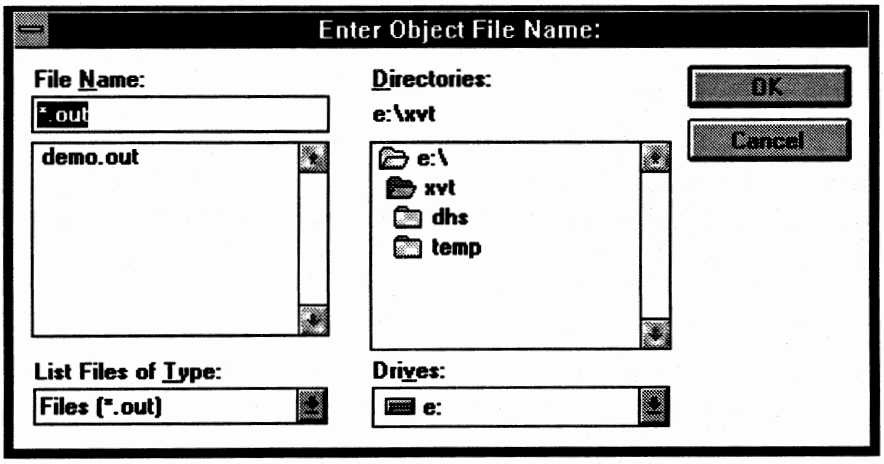
- Motorola S-Record *
- Intel Hex *
- Tekhex *
- HMI Binary *
- Common Object File Format (COFF)
- Software Development Systems (SDS)
- Archimedies Debug Format (IAR)
- IEEE-695
- Extended Tekhex
- Microtec Research Absolute
- Hitachi

  * - No symbol information (including line numbers) supported in this format

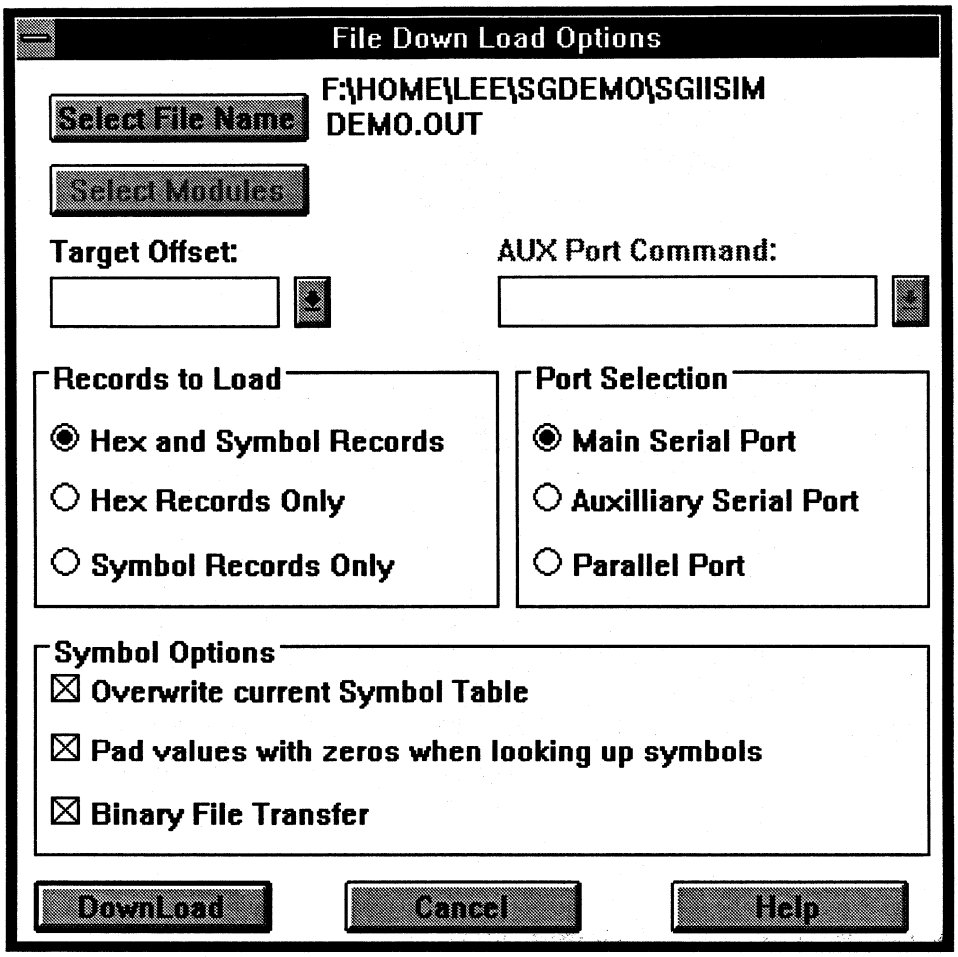
Most compiler/assembler manufacturers provide linkers that can produce one of the above formats. Other formats can be converted to a format that SourceGate II can read through the use of a Converters Disk supplied free of charge by HMI.

Downloading An Object File

To download an object file into memory, choose 'File/Download Hex File'. You will then be prompted for a filename:



At this point, enter the name of the file to download into memory. Remember that the object file must be in one of the previously listed formats. After you select the file, the "File Down Load Options" window will be shown:



After selecting from the options shown in this window, pressing the **DownLoad** button will download the file into either emulation overlay or target system memory depending on how the memory is mapped in the Configuration window.

File Down Load Options Window Fields

This window allows you to define the file to be downloaded using several options. An overview of each of these options is as follows:

'Target Offset' Field

This field allows you to enter an offset value. This number will shift the load address of each record in the file being loaded. For example, if you enter a target offset value of 8, then a record that is specified as loading at 400H will instead load at 408H.

'AUX Port Command' Field
This field is not applicable since the *Lite* has no auxillary serial port.

'Records To Load' Fields
These fields allows you to define what records get loaded. The options are as follows:

Hex and Symbol Records - All records in the file will be read.

Hex Records Only - Only the hex code portion of the file will be read. All symbol information is ignored.

Symbol Records Only - Only the symbol portion of the file will be read. All hex code information is ignored.

'Port Selection' Fields
These fields allow you to define the which port is used for the download. The options are as follows:

Main Serial Port - The file will be downloaded using the main serial port.

Auxiliary Serial Port - This field is not applicable since the *Lite* has no auxillary serial port.

Parallel Port - The file will be downloaded using the parallel port.

'Symbol Options' Fields
If symbols are being downloaded, these options influence their transfer. The options are as follows:

Overwrite Current Symbol Table - Any current symbol table located in memory will be overwritten with the new symbol table information. Otherwise, the new symbol table will be added to the old information.

Pad Values With Zeros When Looking Up Symbols - All symbols read will have zeros padded at the beginning of their address. For example, a symbol with the value of 400H will be stored as 00000400H in memory. This feature can help qualify addresses in the trace and assembly windows which eliminates erroneous listings of symbols in these windows.

Binary File Transfer - All files will be converted to binary before transmitting. This will speed up the file transfer over transmitting in ASCII.

File Down Load Options Window Buttons

The following buttons are available in this window:

Select File Name

This button allows you to specify the path and file to be downloaded.

Select Modules

This button allows you to specify the individual modules to be loaded from a file. This can help speed up file transfers due to the loading of select module symbol information.

NOTE: This is a future enhancement not supported in this release of SourceGate.

DownLoad

This button begins the file downloading process. This button will be greyed out if no filename has been selected.

Cancel

This button cancels the file loading process.

Help

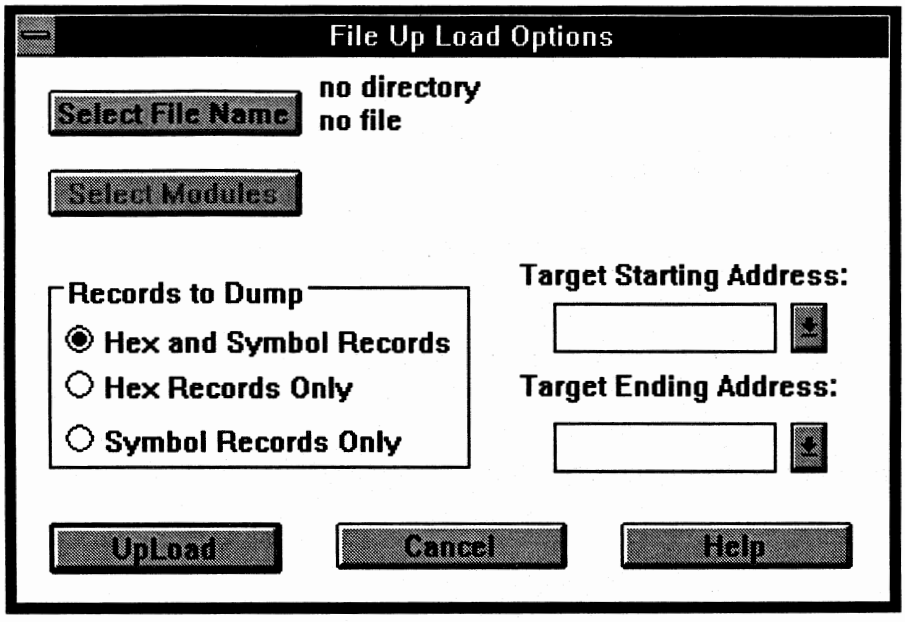
This button invokes help for this window.

Scroll Box

The **Scroll Box** button allows you to choose from previously entered target offset values.

Uploading An Object File

After you have a working version of your software, you probably will want to save it to a file so that you can burn a PROM and then test it in your system. To do this, choose 'File/Upload Hex File' and enter a filename. You will then be prompted for several parameters as shown below:



File Up Load Options Window Fields

A description of the other fields available in this window follows:

Records To Dump

This section consists of three options which determine the information that is saved as part of the dumped file. The following is an overview of each of these options:

> **Hex and Symbol Records** - Dump both hexadecimal code and any symbol information to the designated file.
>
> **Hex Records Only** - Dump only the hexadecimal code to the designated file. Suppress the dump of any symbol information.
>
> **Symbol Records Only** - Dump only the symbol information to the designated file. Suppress the dump of any hexadecimal code.

Target Starting Address

This field defines the beginning address of the memory to be uploaded.

Target Ending Address

This field defines the ending address of the memory to be uploaded.

File Up Load Options Window Buttons

A description of the functions of the buttons available in this window are as follows:

Select File Name

The **Select File Name** button will prompt you for the name of the file to be saved.

Select Modules

The **Select Modules** button allows you to specify from which modules you want symbol information uploaded into the output file.

NOTE: This is a future enhancement not supported in this release of SourceGate.

UpLoad

The **UpLoad** button begins the upload process to the selected file.

Cancel

The **Cancel** button cancels the upload operation and closes this window.

Help

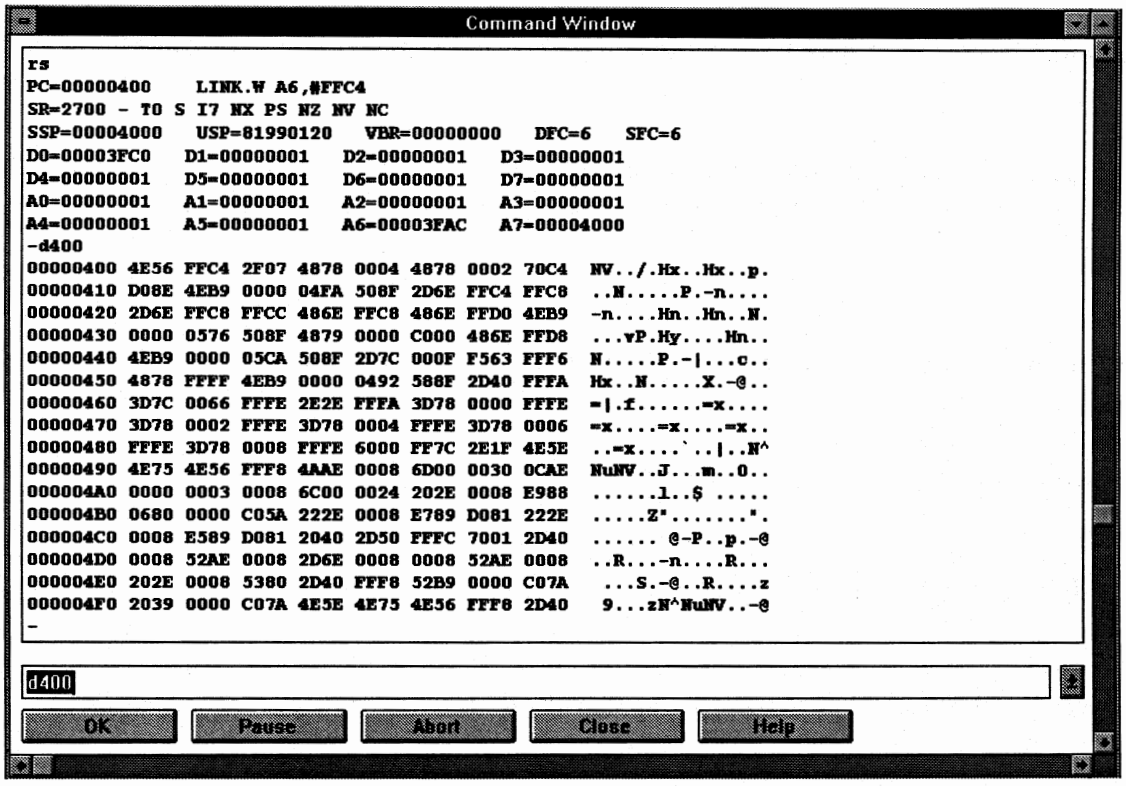
The **Help** button invokes help for this window.

Scroll Box

The **Scroll Box** button allows you to select from previously entered values.

Using The Command Window

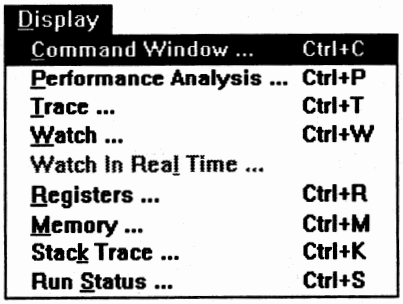
The Command window is the window that you will use to enter commands that affect the emulator. Emulator responses to these commands will also be shown in this window. The Command window has the following format:



An overview of the operation of the Command window follows. **Note: The Command window examples shown in the following sections were obtained from a 68331/332 *Lite* emulator. 68HC16 *Lite* screens may show different data.**

Opening The Command Window

To open the Command window, choose 'Display/Command Window':



An alternate method of opening the Command window is to use the accelerator key combination Ctrl+C.

Command Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button always executes the command displayed in the command entry field.

Pause

The **Pause** button stops the information from scrolling on the screen. After the **Pause** button is pressed, the button will display **Resume**. Pressing the **Resume** button will allow the scrolling to continue.

Abort

The **Abort** button will abort the command that is currently executing.

Close

The **Close** button will close the Command window.

Help

The **Help** button will invoke Command window help.

Scroll Box

The **Scroll Box** button will allow you to scroll through any previously entered commands.

Entering Command Window Commands

All commands are entered in the command entry field located in the Command Window. Once a command is entered, the ENTER key or OK button will initiate execution of the command. Previously entered commands can be recalled by pressing the **Scroll Box** icon located next to the command entry field. A previously entered command can be highlighted using the mouse or the arrow keys. The highlighted command can be then be executed by pressing the ENTER key or OK button.

In the Command Window, the command abbreviation and the first parameter are not required to be separated, but may be separated by one or more spaces. If multiple parameter fields are used, they must be separated by a recognized delimiter. Allowed delimiters include a comma, a single space, or multiple spaces. If a comma is used as the delimiter, it may be preceded and followed by one or more spaces. This allows you some flexibility in the entering of commands.

Command Line Overrides

An override is an optional command parameter that allows you to temporarily override the established global (default) parameter. Overrides are entered on the command line in brackets or parenthesis. More than one override may be specified in the brackets and several may be entered immediately adjacent to each other without separating spaces or delimiters.

Command line overrides do not apply to all commands. They are represented in the command format as a square brackets enclosing the override codes that apply to the command.

S (Supervisor Mode)

Memory addresses given in the command are assumed to be references into either the Supervisor Data or Supervisor Program memory partition. Which one is used will be determined by other overrides or global parameters.

U (User Mode)

Memory addresses given in the command are assumed to be references into either the User Program or User Data memory partition. Which one is used will be determined by other overrides or global parameters.

P (Program Mode)

Memory addresses given in the command are assumed to be references into either the User Program or Supervisor Program memory partition. Which one is used will be determined by other overrides or global parameters.

D (Data Mode)

Memory addresses given in the command are assumed to be references into either the User Data or Supervisor Data memory partition. Which one is used will be determined by other overrides or global parameters.

L (Long Word Mode)

The optional data size of input parameter or output (display) values is selected to be "Long Word".

W (Word Mode)

The optional data size of input parameter or output (display) values is selected to be "Word".

B (Byte Mode)

The optional data size of input parameter or output (display) values is selected to be "Byte".

T (Target System)

Memory will be accessed in the Target System's memory space overriding the memory map as defined in the configuration menu.

R (Repeat)

Rd or Rx

This override will re-execute the current command in three optional formats.

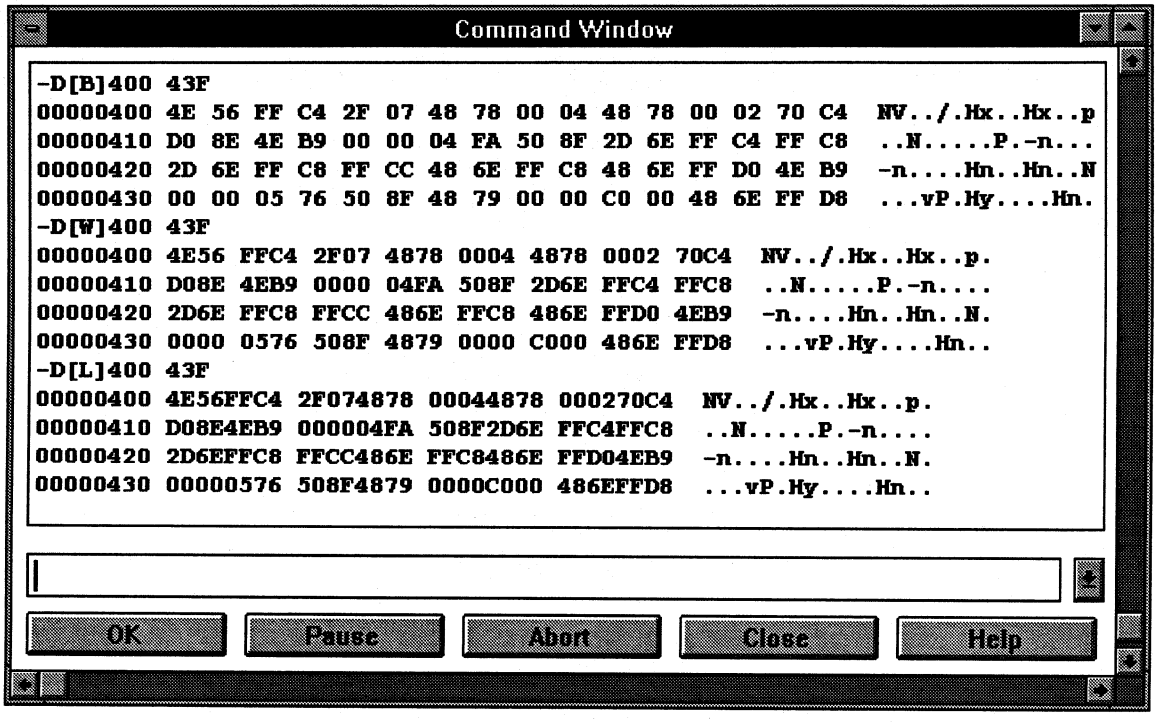
If an R is entered alone, then the command will re-execute each time ENTER is pressed until a period followed by ENTER is pressed.

If an R is entered followed by a number, then the command will re-execute 'd' times.

If an R is entered followed by an X, then the command will be re-executed continually until ENTER is pressed.

Override Example

The following is an example of using the "B", "W", and "L" overrides with the "D" or "Dump" command. Overrides are used similarly with other commands:



Using Symbols In The Command Window

Symbols can be entered in lieu of addresses or data values in Command window commands by typing a colon (:) followed by the symbol name. The value associated with the symbol name will be substituted in the command. Also, a number can be added to or subtracted from a symbol value by following the symbol with "+value" or "-value", respectively, where "value" is the number to be added.

The ':' command is used for symbol table display and editing. The 'symbol', '=', and 'value' portions of this command are all optional. This command comes in many different formats. Each of these is discussed in the sections that follow.

:

':' given by itself on a line causes SourceGate to display the entire contents of its internal symbol table.

:symbol

':' followed by a symbol will cause SourceGate to search for and display the first occurrence of the named symbol in the internal symbol table.

:symbol*

':' followed by a symbol with a wildcard character '*' will cause SourceGate to search for and display all occurrences of the symbol name including wildcards. For example, if ":test*" is entered, SourceGate will display all symbols of this type such as "test", "test1", "testit", etc.

:module/symbol

':' followed by a module name, followed by a '/', followed by a symbol name will cause SourceGate to search for the named symbol in or following the named module in the symbol table. If the symbol is found, it will be displayed.

:symbol=value
:module/symbol=value

These ':' commands perform symbol lookup as outlined above, but when the named symbol is found, the old value of the symbol is discarded, and the given value substituted for it.

'value' may be given as a hexadecimal number, or may be given as the name of another symbol, preceded by a colon. If the named symbol does not exist it will be created. If 'module' is given, the symbol will be created in the named module, otherwise a dummy module name of "USER_DEFINED_MODULE" is assumed.

Using Module Commands In The Command Window

Module commands can be performed while in the Command window. These commands influence the information that is shown in the Command window. The following commands are used to enable a particular level of debugging on a module or globally.

:=A [module] Assembly Mode
:=C [module] Clear Symbol Table
:=D [module] Display Current Modes
:=M [module] Mixed Mode
:=S [module] Source Mode

An explanation of each of these commands is given in the sections that follow.

:=A [module] Assembly Mode Command

The ':=A' command sets the named module (or all modules if none is named) to Assembly Level Debug mode. This is the default mode for all modules when SourceGate is started. While in this mode, only assembly language will be shown in the output of any Command window command, and all source code information will be suppressed.

:=C [module] Clear Symbol Table Command

The ':=C' command clears all symbols from the internal symbol table for the named module (or all modules if none is named). The symbol information for the cleared module must be read back into the internal symbol file in order to perform symbolic/source debugging on that module.

:=D [module] Display Current Modes Command

The ':=D' command displays source debug information for the named module (or all modules if none is named). Source debug information consists of the line number map acquired by SourceGate through examination of the module's line number symbols (whose format is set with the SourceGate command line switch '-L' or through the Source Code Interface Options window), the module's source code file name, and the current debug mode set for that module (Assembly, Mixed, or Source-only).

:=M [module] Mixed Mode Command

The ':=M' command sets the named module (or all modules if none is named) to Mixed Debug mode. Once a module has been enabled for mixed debug mode, that module's source file will be opened, and the source lines for that module will be inserted at appropriate points into the output of all commands which disassemble code (i.e. the 'L' (List) command, 'SS' (single-step) command, etc.).

When the ':=M' command is used, SourceGate must be able to locate and successfully open each named module's source code file. These source code files are assumed to reside in the current directory, and their names are assumed to be the module's name, with the extent of ".C".

If any of these assumptions are false, SourceGate will prompt for a source code file name, giving the name of the module for which it cannot find the source code file. The full path name of the file, along with the corresponding drive designator, should be entered at this time.

If no source code file is available, entering no file name at the prompt will cause SourceGate to set that module back to Assembly mode debug (which requires no source code file), and continue on to the next module to be set to Mixed debug mode.

:=S [module] Source Mode Command
The ':=S' command sets the named module (or all modules if none is named) to Source-only Debug mode. When source-only debug mode is selected for a particular module, all assembly language displays for that module will be suppressed. (Some assembly may be displayed in the source only mode. This assembly is from the library routines of the compiler and is source code to the emulator since most of these routines are written in assembly language). In addition, that module's source file will be opened and the source lines for that module will be inserted into the output of all commands which disassemble code (i.e. the 'L' (list) command, 'SS' (Single-Step) command, etc.).

When the ':=S' command is used, SourceGate must be able to locate and successfully open each named module's source code file. These source code files are assumed to reside in the current directory, and their names are assumed to be the module's name, with the extent of ".C".

If any of these assumptions are false, SourceGate will prompt for a source code file name, giving the name of the module for which it cannot find the source code file. The full path name of the file, along with the corresponding drive designator, should be entered at this time.

If no source code file is available, entering no file name at the prompt will cause SourceGate to set that module back to Assembly mode debug (which requires no source code file), and continue on to the next module to be set to Source debug mode.

Creating Automated Command Files From The Command Window

To begin recording an automated command session file that can be played back later, choose "File/Record". This activates the Command window and prompts you to enter the name of the file in which to store the recorded information. After you have entered the required filename, any commands that are typed in the Command window are recorded in the file until you choose "File/Stop Recording". The file is then closed and can be played back by choosing "File/Execute". **Note: This feature only is supported in the Command window and will not record operations in other windows.**

Available Command Window Commands

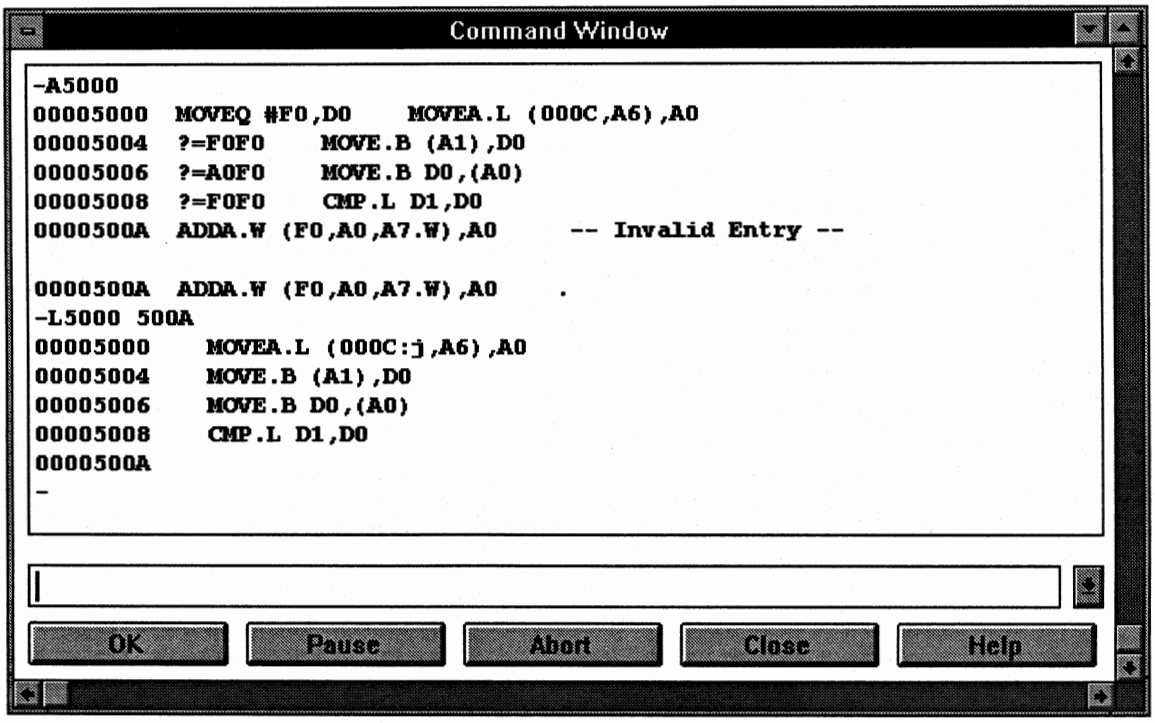
The following commands are available from the command line for this emulator:

A (In-Line Assembly)
BM (Byte Mode)
BSD(+/-) (Tri-State Data Bus Buffers During DMA) (68340 Only)
BTE (Set 8-Bit Boot Chip Select) (68340 Only)
C (Compare Memory Block)
CM (Show Status Of Control Board Memory)
CONFIG (Recall Factory Configuration)
CONFIGT (Recall Factory Configuration For Target)
CRL (Load Configuration Registers)
D (Dump Memory Block)
DDP (Dump Dual Port Memory Block)
DM (Data Memory Mode)
DW (Write Line Active For Target Writes Only)
E/EN (Examine/Change Memory)
EDP/EDPN (Examine/Change Dual Port Memory)
EDT (Events Disabled For Temp. Breakpoints)
EET (Events Enabled For Temp. Breakpoints)
ET (Show Status For EDT/EET)
EW (Write Line Active For All Memory Writes)
EXP (Examine/Change Peripheral)
EXR (Examine/Change Peripheral Register)
F (Fill Memory Block)
FDP (Fill Dual Port Memory Block)
G (Begin Real-Time Emulation)
HLD (Hex Listing Disabled)
HLE (Hex Listing Enabled)
I/IN (Input)
L (List Disassembled Code)
LM (Long Word Display Mode)
M (Move Memory Block)
MMC (Check Configuration Menu Setup)
MT/MTL (Test Block Of Memory)
O/ON (Output)
PM (Program Memory Mode)
RD/RDP (Read Object File Into Memory)
RD(%) (Read Configuration File)
RD(!) (Read Firmware File Into Flash ROM)
RP (Reset Processor, Continue Emulation)
RS (Reset Processor, Stop Emulation)

SH (Show Status For SHD/SHE)
SHD (Serial Handshake Disabled)
SHE (Serial Handshake Enabled)
SM (Supervisor Memory Mode)
SP (Stop Emulation)
SR/SRN (Search Block Of Memory)
SS (Single Step)
ST (Show Emulator Status)
UM (User Memory Mode)
VASM (View Assembly)
VER (Display Firmware Version)
W (Write Memory Contents To File)
WM (Word Display Mode)
X (Examine/Change Registers)

All of the preceding commands are described in detail in the sections that follow. Optional parameters are shown enclosed by parenthesis.

A (In-Line Assembly) Command



This command allows assembly language mnemonics to be entered into memory starting at the specified address (In-Line Assembly). Once the Assembly command is entered, the address will be displayed with the cursor to the right. At this point, CPU32 assembly language instructions can be entered. After each entry, the emulator displays the next address and prompts for the next instruction. An error in the entered instruction will be flagged by displaying "-- Invalid Entry--" and re-displaying the current address. A period (.) followed by a carriage return terminates the entry sequence and the command.

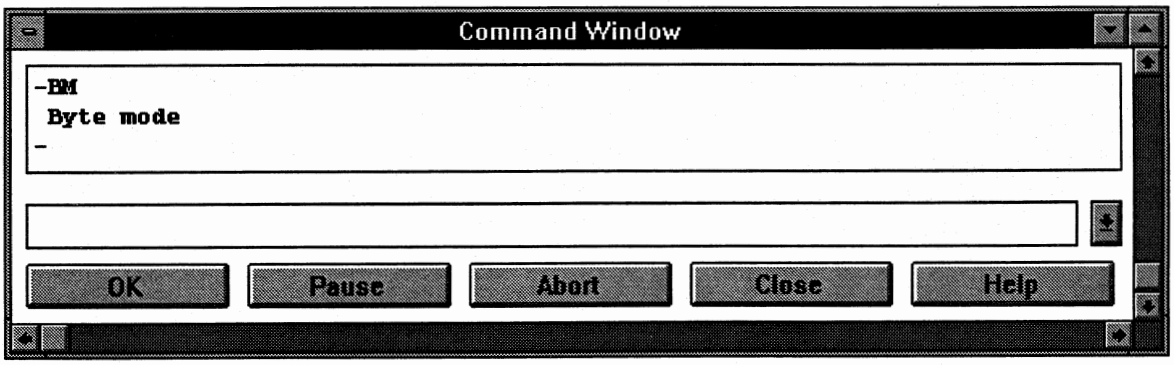
The format for the Assemble command is:

A([SUPDT]) Addr

where:

Addr:	is the hexadecimal starting address
SUPDT:	are the available overrides

BM (Byte Mode) Command



This command sets the default for input or output values to "Byte Mode". While in Byte mode, commands such as D (Dump memory) will display values in byte mode instead of word mode or long word mode, which are the other choices (see WM and LM).

In addition, commands which require an input value will be affected by this command. For example, if the Fill command, F 8000 9000 33 is executed while in byte mode, each byte of memory from 8000H through 9000H will contain 33H. If this same command was executed with the emulator in word mode, then 0033H would be stored in each word location from 8000H to 9000H. If this same command was executed with the emulator in long word mode, then 00000033H would be stored in each long word location from 8000H to 9000H.

BSD(+/-) (Tri-State Data Bus Buffers During DMA) Command (68340 Only)

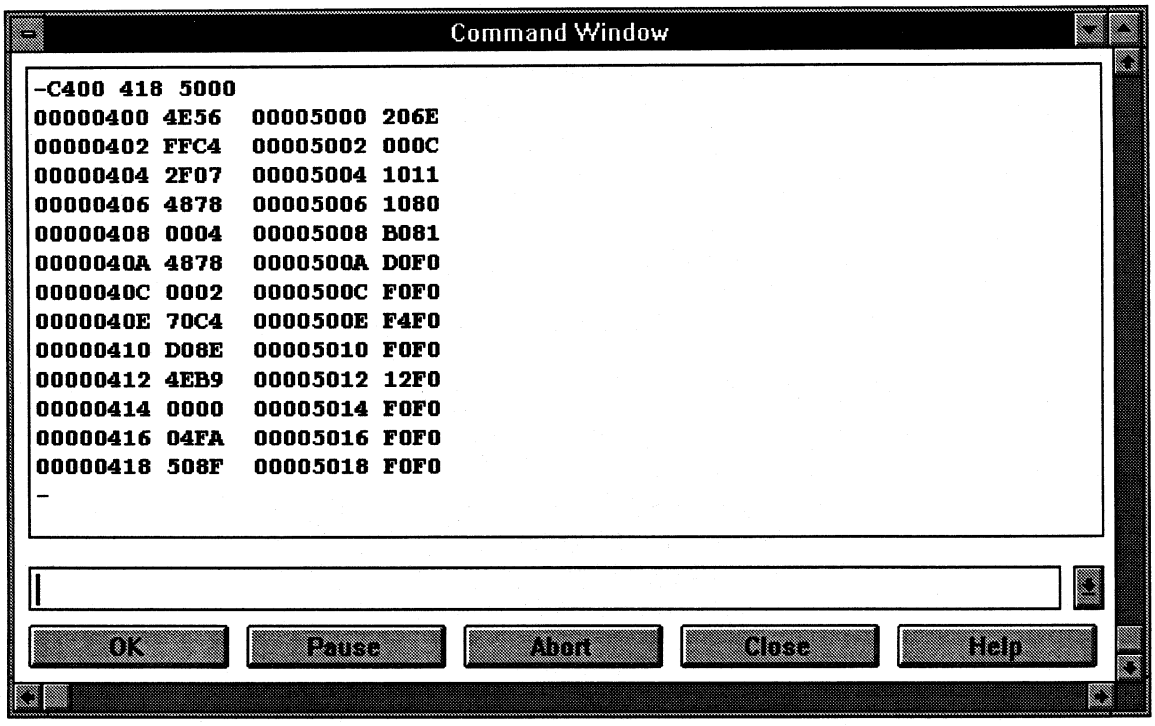
The BSD(+/-) command allows the user to tri-state the *Lite's* data bus buffers during DMA operations. This may be necessary for Single Address Mode DMA to be supported correctly.

To allow data bus tri-stating, the BSD+ command should be used. To disallow data bus tri-stating, the BSD- command should be used. BSD- is the default configuration.

BTE (Set 8-Bit Boot Chip Select) Command (68340 Only)

The BTE command will program CS0' to be an 8-bit chip select value. This condition will be reflected in the Configuration meun. Programming the address mask register for 16-bit operation or executing the CONFIG command will set CS0' back to a 16-bit chip select.

C (Compare Memory Block) Command



The Compare command will compare one block of memory to another. The first word in the first block will be compared to the first word in the second block, and so on. Any non-comparisons will be displayed on the screen showing the address in the first block with its data value and the address in the second block with its data value. If the two blocks match, then nothing will be displayed. **Note that you can also use the Memory window to compare memory.**

The format for the Compare command is:

C([SUPDTR]) Baddr1 Eaddr1 ([SUPDT])Baddr2

where:

Baddr1:	is the hexadecimal beginning address of the first compare block
Eaddr1:	is the hexadecimal ending address of the first compare block
Baddr2:	is the hexadecimal beginning address of the second compare block
SUPDTR:	are the available overrides

The size of the first block sets the number of words to be compared. For each word in block one that does not match the corresponding word in block two, a line is displayed on the screen.

This display line has the format:

Addr1 Value1 Addr2 Value2

where:

Addr1:	is the address of the word in block one
Value1:	is the contents of addr1
Addr2:	is the address of the word in block two
Value2:	is the contents of addr2

CM (Show Status Of Control Board Memory) Command

Executing this command will show the emulator options that are installed. It should be noted that "System RAM" in this case refers to the system RAM inside the emulator (not the PC) and "Dual Port RAM" refers to the 2K of DPRAM used to watch variables during emulation.

CONFIG (Recall Factory Configuration) Command

This command will return the emulator menus, ports, memory map, predefined sequences and other battery backed-up configuration values to the factory configuration.

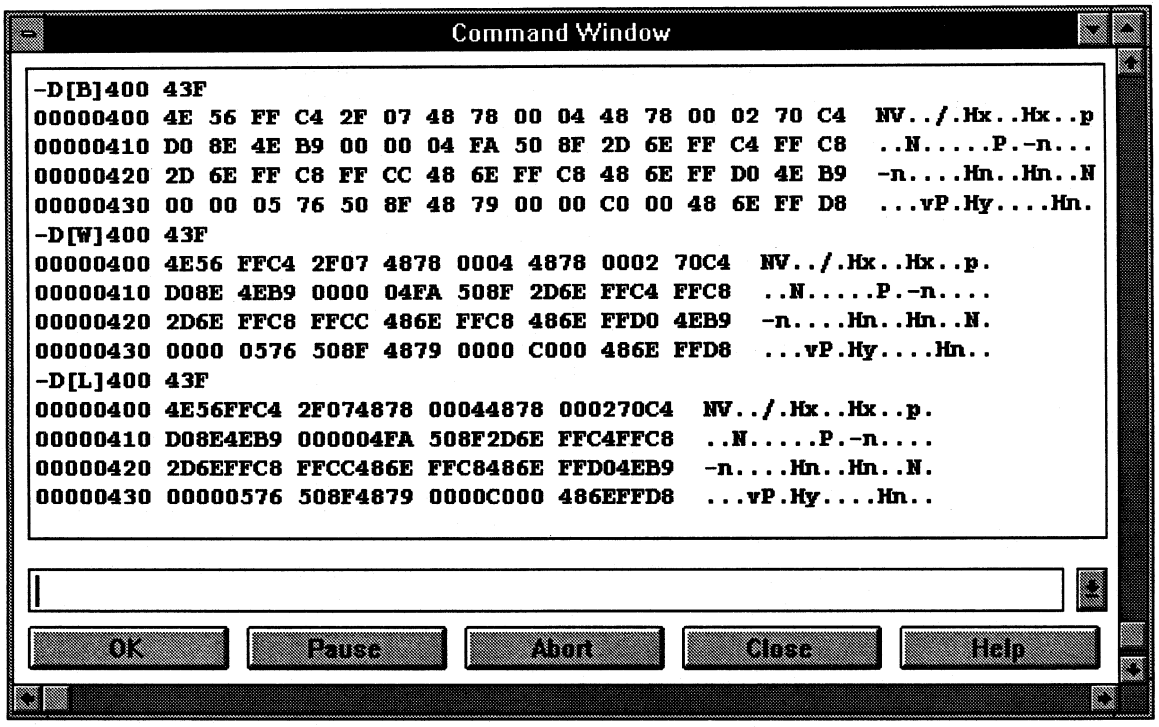
CONFIGT (Recall Factory Configuration For Target) Command

This command will return the emulator menus, memory map, predefined sequences, and other battery backed- up configuration values to the factory configuration for target system operation. When this command is executed, all parameters that can be set to come from the target system (clock source, memory, etc.) will be configured to come from the target system.

CRL (Load Configuration Registers) Command

This command will read the current values of the registers shown in the Configuration window and will save these values in battery-backed memory. Then, each time a CONFIG command is issued, these values will be re-read from memory and programmed into the processor and the Configuration window.

D (Dump Memory Block) Command



The Dump Command will display the contents of memory in the specified address range showing the hexadecimal data with its ASCII equivalent on the screen. If no ending address is given, then one screen of data will be displayed and the command will terminate. If no starting address is specified then the last ending address (whether user entered or default) will be used as the starting address. Using this method of defaulting allows you to dump a display of memory from one starting address and get the next and subsequent pages of data dumped by simply typing the "D" and a carriage return. **Note that you can also use the Memory window to display the contents of memory.**

The Format for the Dump Command is:

D([SUPDLWBTR]) (Baddr) (Eaddr)

where:

Baddr:	is the hexadecimal beginning address of the block of memory to be displayed
Eaddr:	is the hexadecimal ending address of the block of memory to be displayed
SUPDLWBTR:	are the available overrides

DDP (Dump Dual Port Memory Block) Command

The DDP Command will display the contents of dual port memory in the specified address range showing the hexadecimal data with its ASCII equivalent on the screen.

If no ending address is given, then one screen of data will be displayed and the command will terminate. If no starting address is specified then the last ending address (whether user entered or default) will be used as the starting address. Using this method of defaulting allows the user to dump a display of memory from one starting address and get the next and subsequent pages of data dumped by simply typing "DDP" and a carriage return.

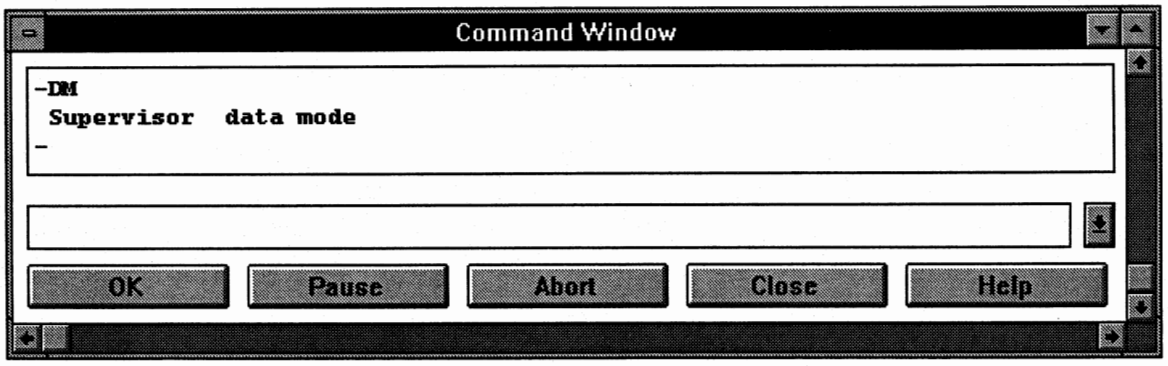
The Format for the DDP Command is:

DDP([SUPDLWB]) (Baddr) (Eaddr)

where:

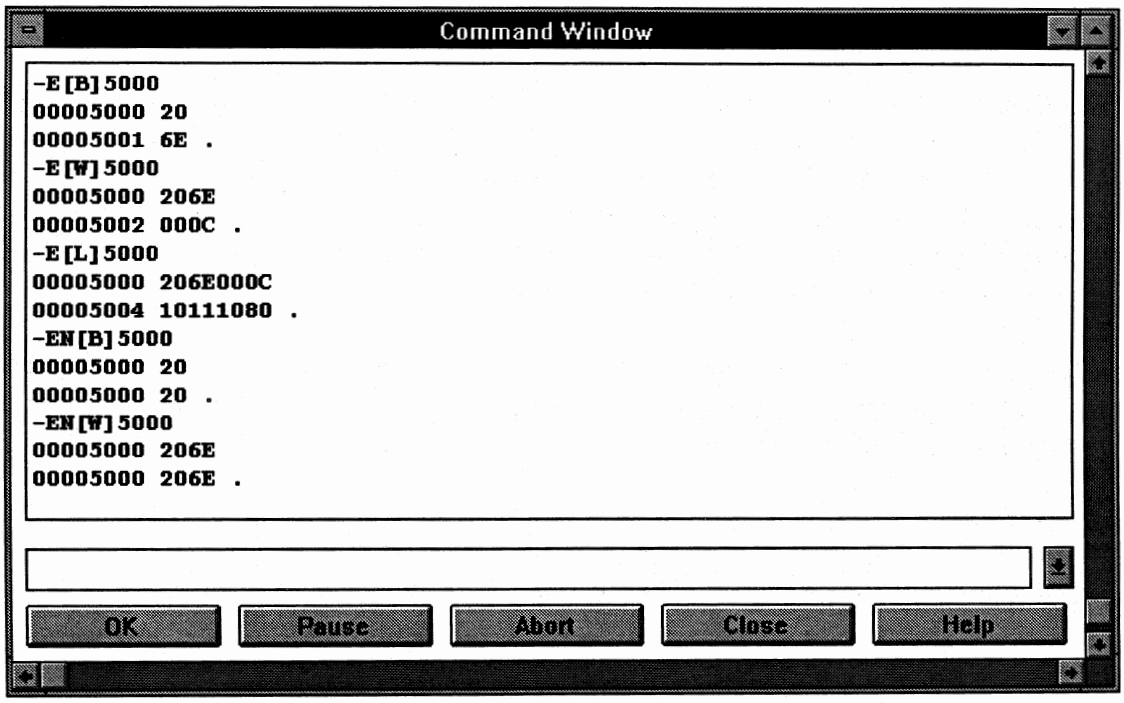
Baddr:	is the hexadecimal beginning address of the block of memory to be displayed
Eaddr:	is the hexadecimal ending address of the block of memory to be displayed
SUPDLWB:	are the available overrides

DM (Data Memory Mode) Command



The execution of this command sets the default memory category to be Data Memory mode. Using the SM (Supervisor Mode) or UM (User Mode) commands further define the default memory mode.

E/EN (Examine/Change Memory) Command



The Enter Command is used to examine memory at the specified location with the option to change its contents. After executing the Enter Command, the selected address will appear on the screen with the current data. The data can be altered by entering new data and pressing a carriage return, or it can remain unchanged by just typing a carriage return. In either case the next address will be displayed with its data. This command can be terminated by typing a period (.) in the data field followed by a carriage return. **Note that you can also use the Memory window to examine and change the contents of memory.**

There is a variation of the Enter command (EN) that does not advance to the next address. Whether you enter a change and a carriage return or just a carriage return, it will display the value at the specified address only. This feature is very useful in monitoring a memory mapped I/O location.

The Format for the Enter Command is:

E([SUPDLWBT]) Addr
or
EN([SUPDLWBT]) Addr

where:

Addr:	is the hexadecimal address of the data value to be displayed for possible substitution
SUPDLWBT:	are the available overrides

EDP/EDPN (Examine/Change Dual Port Memory) Command

The EDP Command is used to examine dual port memory at the specified location with the option to change its contents. After executing the EDP Command, the selected address will appear on the screen with the current data. The data can be altered by entering new data and pressing a carriage return, or it can remain unchanged by just typing a carriage return. In either case the next address will be displayed with its data. This command can be terminated by typing a period (.) in the data field followed by a carriage return.

There is a variation of the EDP command (EDPN) that does not advance to the next address. Whether you enter a change and a carriage return, or just a carriage return, it will display the value at the specified address only.

The Format for the EDP Command is:

EDP([SUPDLWB]) Addr

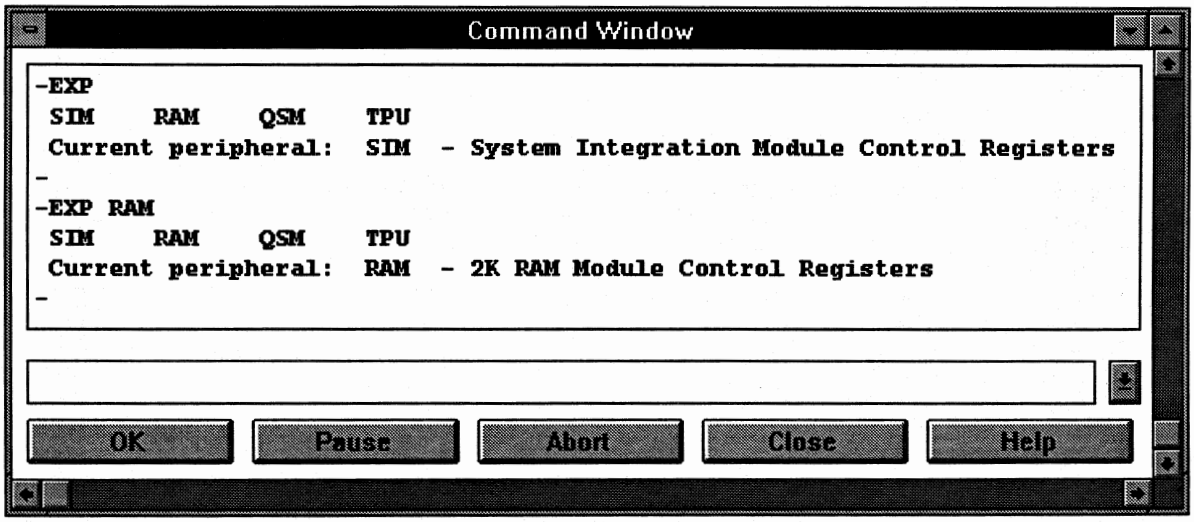
or

EDPN([SUPDLWB]) Addr

where:

Addr:	is the hexadecimal address of the data value to be displayed for possible substitution
SUPDLWB:	are the available overrides

EXP (Examine/Change Peripheral) Command



The EXP command changes the current default peripheral. The EXR command can then be used to examine/change a specific register within the default peripheral. Entering EXP by itself on the command line will show the current default peripheral.

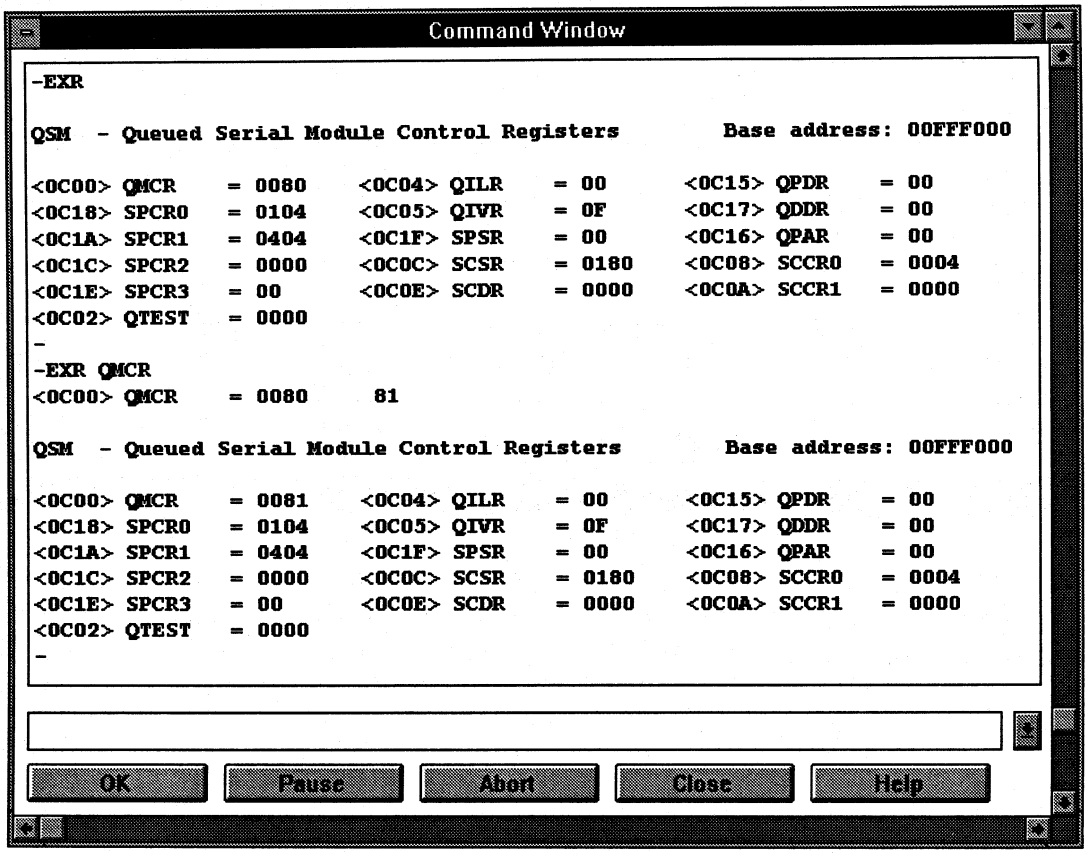
The format of the EXP command is:

EXP per

Where:

per is the desired default peripheral (QSM, RAM, SIM, or TPU)

EXR (Examine/Change Peripheral Register) Command



The EXR command allows a register in the current default peripheral to be examined/changed. The default peripheral can be selected by using the EXP command. Entering EXR by itself on the command line will show all registers associated with the current default peripheral.

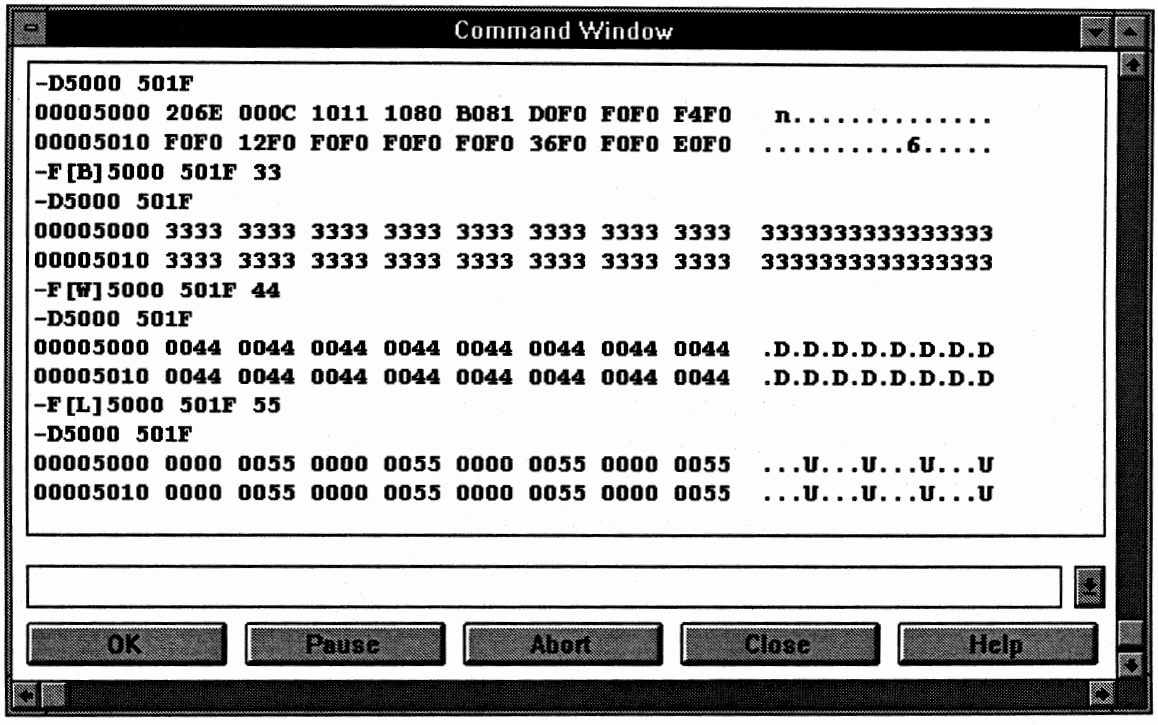
The format of the EXR command is:

EXR reg

Where:

reg is the register peripheral to be examined or changed

F (Fill Memory Block) Command



The Fill Command is used to fill memory with a constant. The given data value will be written to each memory location within the specified range. The value may be a byte or a word in length, determined by the global parameters and overrides used. **Note that you can also use the Memory window to fill memory.**

The format for the Fill Command is:

F([SUPDLWBTR]) Baddr Eaddr Val

Where:

Baddr:	is the hexadecimal beginning address of the memory block to be filled
Eaddr:	is the hexadecimal ending address of the memory block to be filled
Val:	Constant value to be used to fill the memory block
SUPDLWBTR:	are the available overrides

FDP (Fill Dual Port Memory Block) Command

The FDP Command is used to fill dual port memory with a constant. The given data value will be written to each memory location within the specified range. The value may be a byte or a word in length, determined by the global parameters and overrides used.

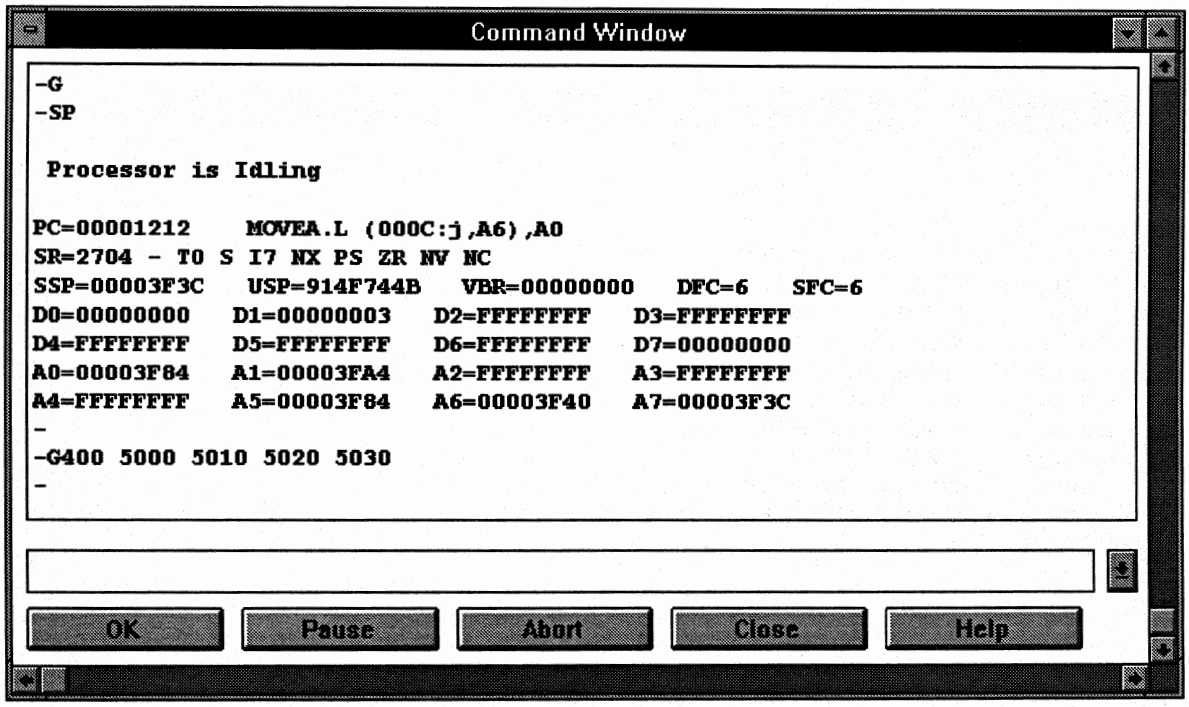
The format for the FDP Command is:

FDP([SUPDLWBTR]) Baddr Eaddr Val

Where:

Baddr:	is the hexadecimal beginning address of the memory block to be filled
Eaddr:	is the hexadecimal ending address of the memory block to be filled
Val:	Constant value to be used to fill the memory block
SUPDLWBTR:	are the available overrides

G (Begin Real-Time Emulation) Command



This command is used to start real-time emulation. The options for this command are to run with temporary breakpoints as set in this command, or without a breakpoint, in which case any break conditions which occur are due to the configuration of the events from the Event windows and the breakpoint sequence as defined in the Sequence Break Point window. If the option for running with breakpoints from this command is selected, then it is important to note that the configuration of the Event and the Sequence windows become inactive during execution of this command. In either case emulation will begin at the start address given or default to the current program counter. If a starting address is to be used it must be immediately adjacent to the G (otherwise it will be taken as a breakpoint address). **Note that you can also use the 'Source/Run' menu item along with the "Run Status" window to begin emulation.**

The format for utilizing the Event and Sequence menu configurations is:

G(Saddr)

The format for setting temporary breakpoints is:

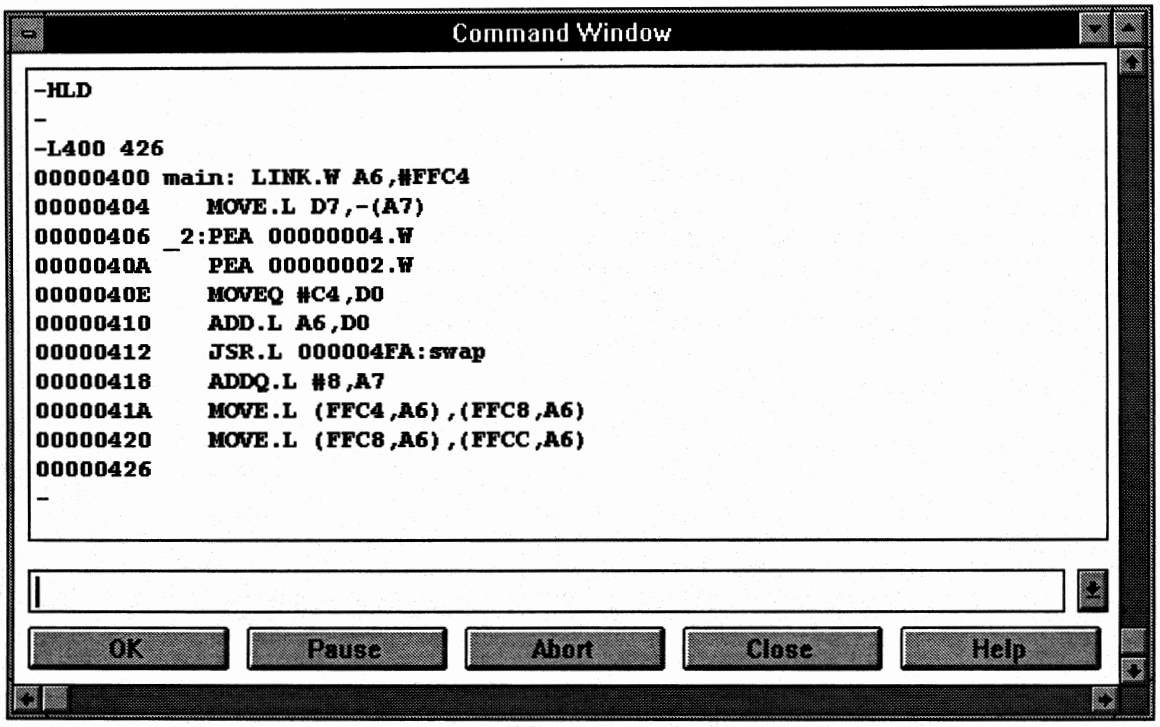
G(Saddr) Braddr1 (Braddr2 Braddr3 Braddr4)

Where:

Saddr: is the hexadecimal address where execution (real-time emulation) will begin

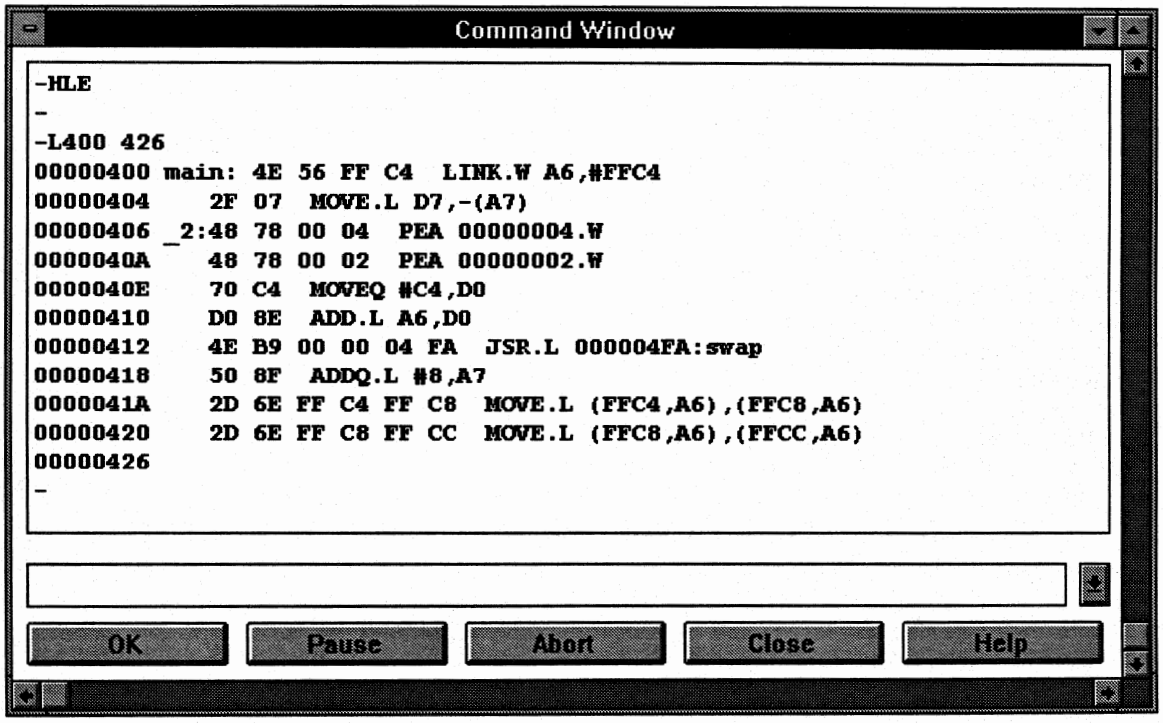
Braddr1-4: are hexadecimal addresses where execution (real-time emulation) will stop (or break) on op-code execution

HLD (Hex Listing Disabled) Command



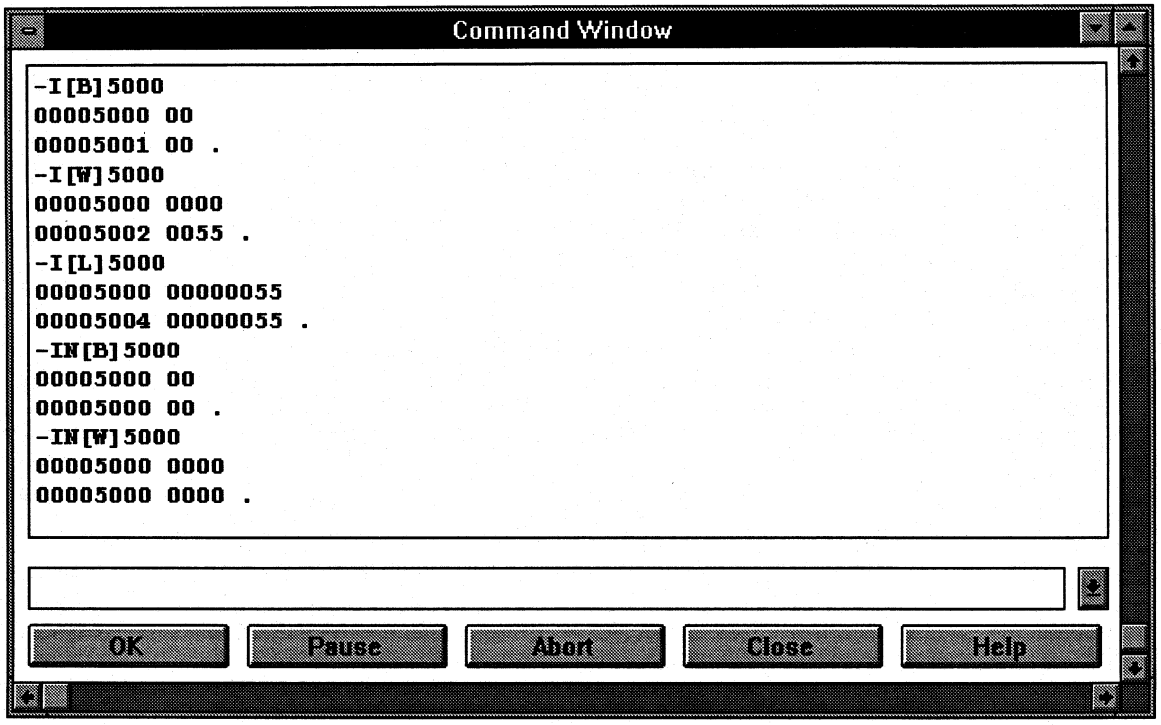
Executing this command will disable the hex information from being displayed with the disassembly information during the L (List) command. To enable the hex information, execute the HLE (Hex Listing Enabled) command.

HLE (Hex Listing Enabled) Command



Executing this command will enable the hex information to be displayed with the disassembly information during the L (List) command. To disable the hex information, execute the HLD (Hex Listing Disabled) command.

I/IN (Input) Command



The I command is used to read and display a certain memory address. This function is very useful when monitoring memory mapped I/O locations. Each time a <cr> is entered, the memory location is read, shown, and incremented to point at the next location. If you wish to suppress the incrementing of the address every time a <cr> is pressed, the IN command should be used. **Note that you can also use the Memory window to read and display memory.**

The format for the Input command is:

I([SUPDLWBT]) Addr

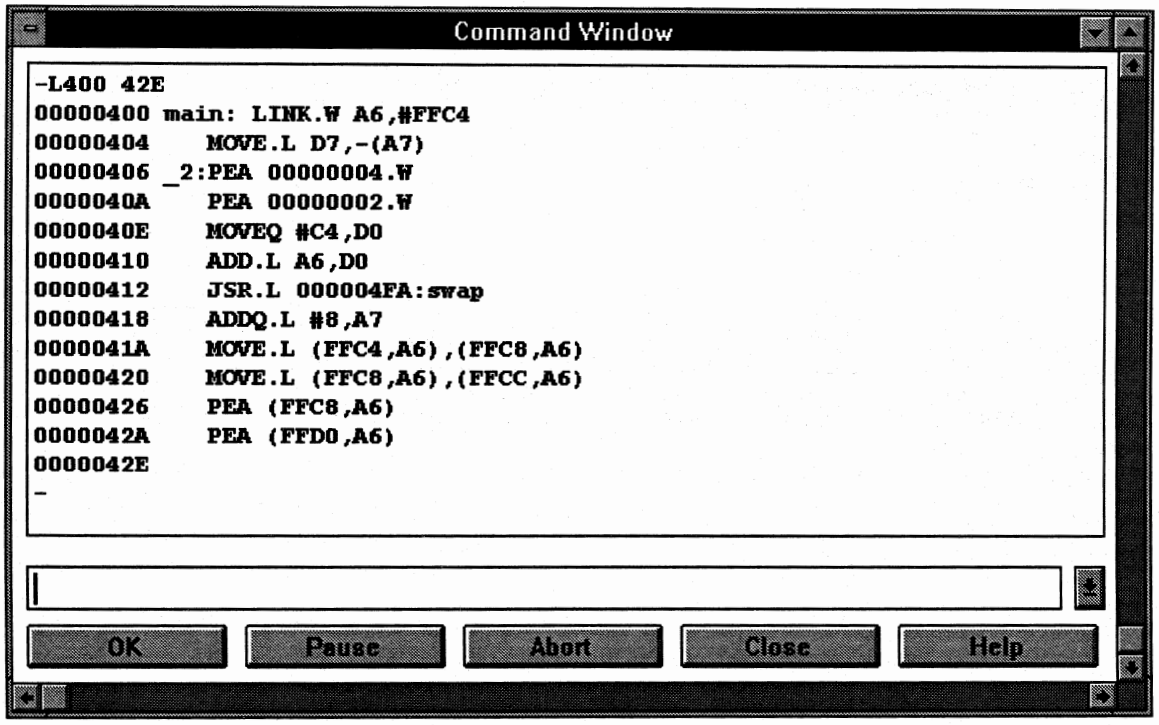
or

IN([SUPDLWBT]) Addr

Where:

Addr: is the hexadecimal address of the data value to be read
SUPDLWBT: are the available overrides

L (List Disassembled Code) Command



This command is used to disassemble the program memory data into assembly language mnemonics. All data and addresses are displayed in hexadecimal. If no ending address is specified, then one screen of disassembled mnemonics will be displayed and the command will terminate. If no beginning address is specified then the last ending address (whether entered or default) will be used as the beginning address. This allows you to disassemble code one display screen at a time by entering the command and an initial starting address for the first display and for the next and each subsequent page only the "L" and a carriage return need be entered. If a range is given, disassembled mnemonics will be displayed from the starting address to the ending address, scrolling the screen if necessary. To pause a scrolling screen enter CNTRL-S and, to resume scrolling a CNTRL-Q. This can also be accomplished by using the "pause" key to stop scrolling and <cr> to resume scrolling. To terminate the command while scrolling depress the space bar or enter a carriage return.

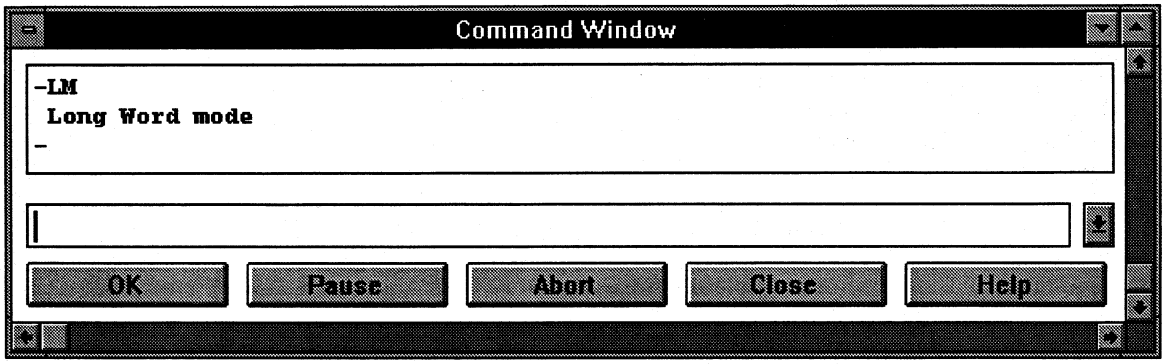
The format for the List Command is:

L([SUPDTR]) (Baddr) (Eaddr)

Where:

Baddr:	is the hexadecimal beginning address of the memory data to be disassembled
Eaddr:	is the hexadecimal ending address of the memory to be disassembled
SUPDTR:	are the available overrides

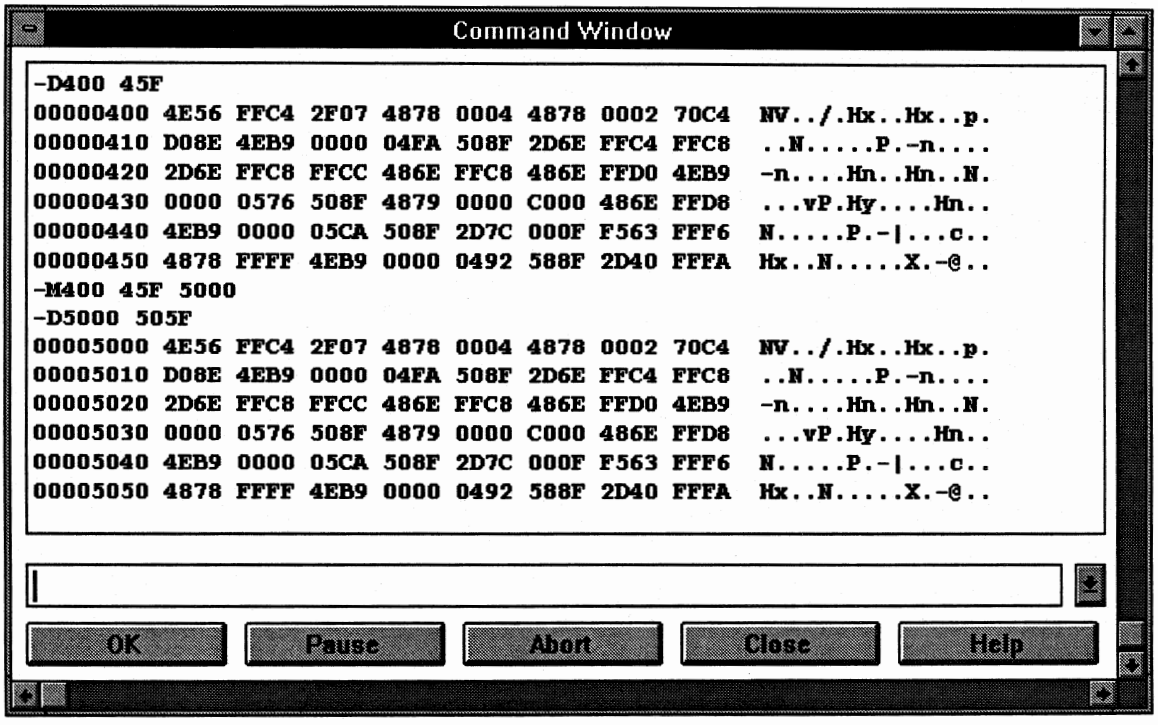
LM (Long Word Display Mode) Command



This command sets the default for input or output values to "Long Word Mode". While in Long Word mode, commands such as D (Dump memory) will display values in Long Word mode instead of word mode or byte mode, which are the other choices (see WM and BM).

In addition, commands which require an input value will be affected by this command. For example, if the Fill command, F 8000 9000 33 is executed while in Long Word mode, each long word of memory from 8000H through 9000H will contain 00000033H. If this same command was executed with the emulator in word mode, then 0033H would be stored in each word location from 8000H to 9000H. If this same command was executed with the emulator in byte mode, then 33H would be stored in each byte location from 8000H to 9000H.

M (Move Memory Block) Command



The Move Command is used to copy the contents of one block to the contents of another block (non- destructive move). Block one is defined with its starting and ending addresses being the first two address fields displayed, and block two is defined with its starting address being the third address field. The contents of the first memory location of block one is copied to the first memory location of block two, and so on. The next prompt character is not displayed until the move is complete. **Note that you can also use the Memory window to move memory.**

The format of the Move Command is:

M([SUPDTR]) Baddr1 Eaddr1 ([SUPDT])Baddr2

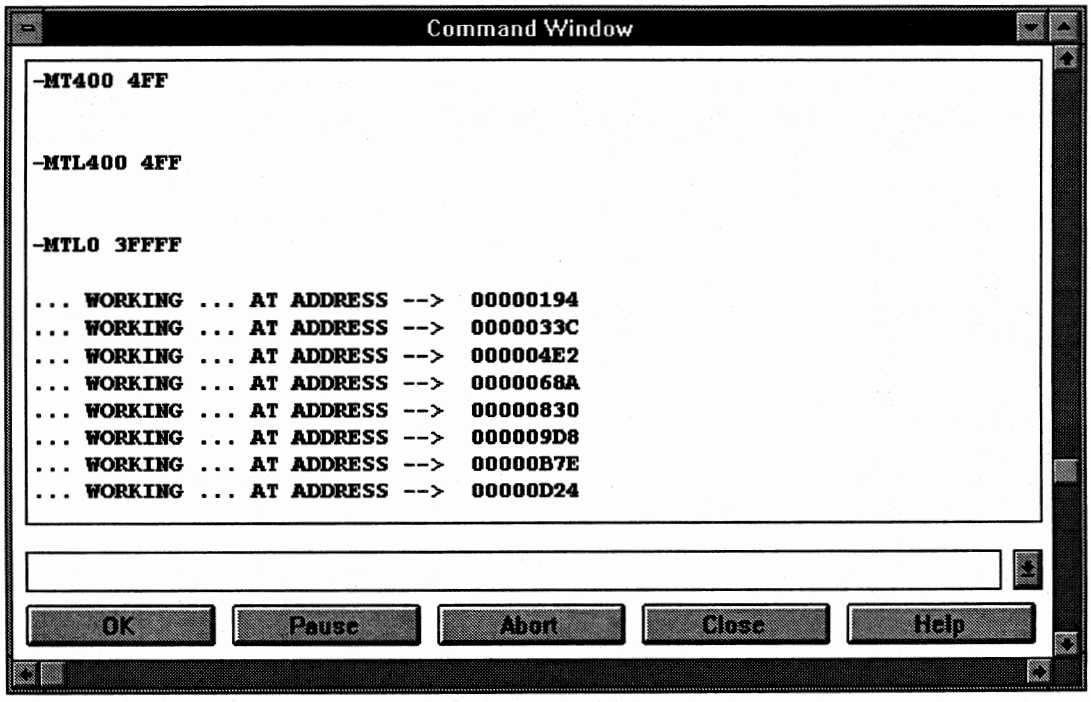
where:

Baddr1:	is the hexadecimal beginning address of the data "Source" block
Eaddr1:	is the hexadecimal ending address of the data "Source" block
Baddr2:	is the hexadecimal beginning address of the data "Destination" block . The size of the "Source" block defines the number of bytes to be copied into the "Destination" block.
SUPDTR:	are the available overrides

MMC (Check Configuration Menu Setup) Command

The MMC command performs a check of the Configuration menu setup and reports if any errors are found. This command is useful for finding improper activator setups without having to go to the Configuration menu.

MT/MTL (Test Block Of Memory) Command


Command Window
-MT400 4FF
-MTL400 4FF
-MTL0 3FFFF
... WORKING ... AT ADDRESS --> 00000194
... WORKING ... AT ADDRESS --> 0000033C
... WORKING ... AT ADDRESS --> 000004E2
... WORKING ... AT ADDRESS --> 0000068A
... WORKING ... AT ADDRESS --> 00000830
... WORKING ... AT ADDRESS --> 000009D8
... WORKING ... AT ADDRESS --> 00000B7E
... WORKING ... AT ADDRESS --> 00000D24
OK
Pause
Abort
Close
Help

This command can be used to test memory (RAM) within the given address range. The short memory test (MT) writes, reads and compares two separate bit patterns for each byte within the range. The bit patterns are 55H and AAH, thus cycling all bits of the byte through both one and zero. The long memory test (MTL) is a walking bit pattern. This test writes, reads and compares the byte while having only one bit in the byte on at a time. This is done by using the Hexadecimal values 1, 2, 4, 8, 10, 20, 40, 80 for each byte within the range. In either test, any read not comparing with what was written into a location will cause the address, data written and data read to be displayed on the next line of the screen. Both of these tests are non- destructive and can be performed without corrupting data currently in memory. A <cr> will terminate this test at any time. **Note that you can also use the Memory window to test memory.**

The format for the Memory Test command is:

MT([SUPDTR]) Baddr Eaddr

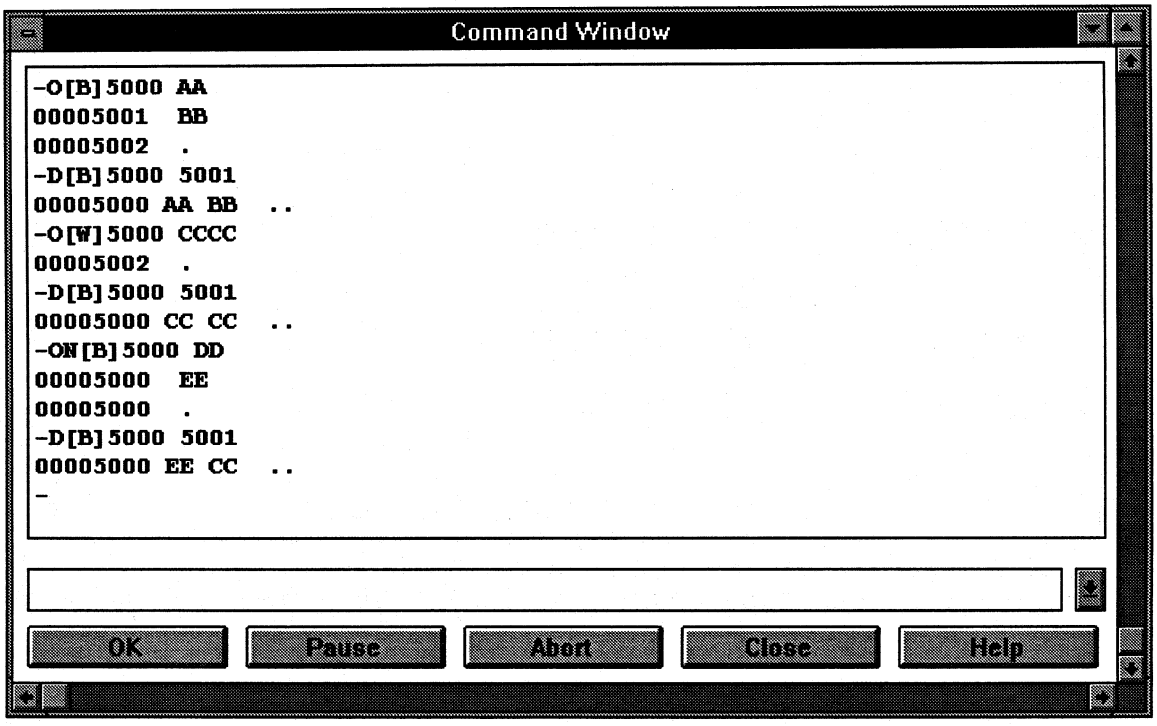
The format for the long Memory Test command is:

MTL([SUPDTR]) Baddr Eaddr

Where:

Baddr:	is the hexadecimal beginning address of the memory block to be tested.
Eaddr:	is the hexadecimal ending address of the memory block to be tested.
SUPDTR:	are the available overrides

O/ON (Output) Command



The O command is used to write to a certain memory address. This function is very useful when testing memory mapped I/O locations. Each time a <cr> is entered, the desired value is written to the displayed address and the address is incremented. The ON command will not increment the address to be written but will write the same address each time a <cr> is entered. **Note that you can also use the Memory window to output the contents of memory.**

The format for the Output command is:

O([SUPDLWBT]) Addr,Val

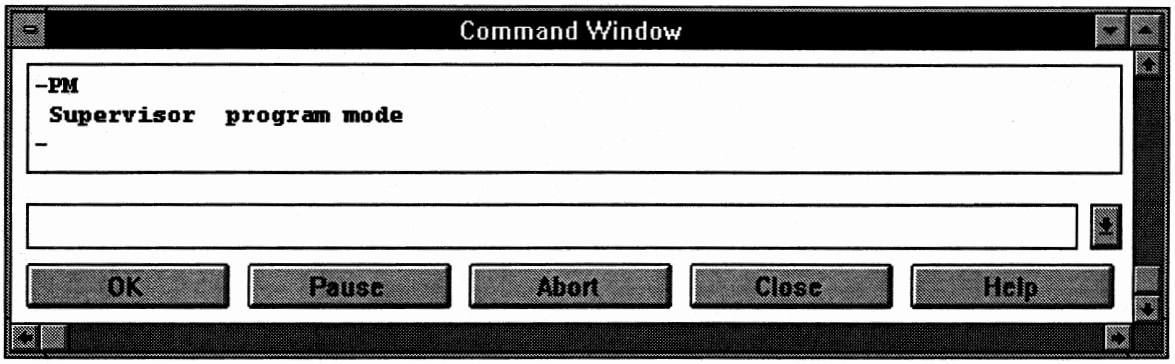
or

ON([SUPDLWBT]) Addr,Val

Where:

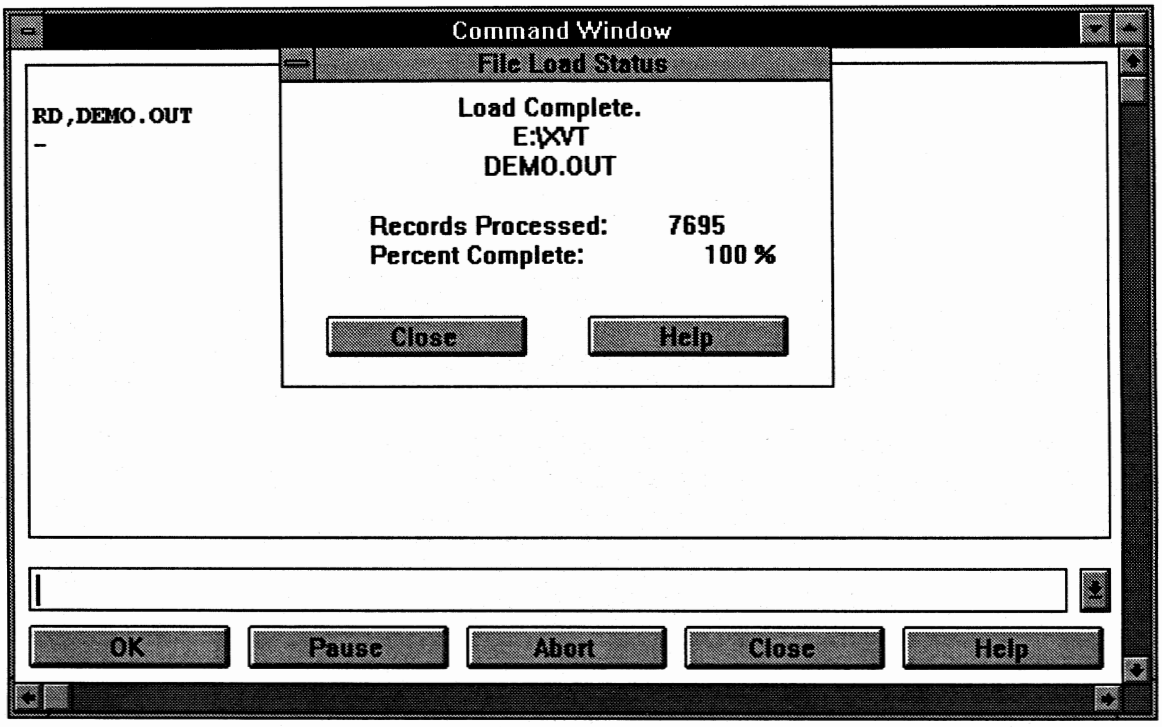
Addr: is the hexadecimal address to be written with the defined value
Val: is the hexadecimal value to be written
SUPDLWBT: are the available overrides

PM (Program Memory Mode) Command



The execution of this command sets the default memory category to be Program Memory mode. Using the SM (Supervisor Mode) or UM (User Mode) commands further define the default memory mode.

RD/RDP (Read Object File Into Memory) Command



This command is used to transfer a hexadecimal file from a host computer to the emulation memory. The file name can be specified with this command. When downloading a file using SourceGate II, the ASCII file is converted to binary before being transferred to the emulator. This essentially doubles the speed of the transfer over the normal ASCII transfer. When downloading from a host which doesn't use SourceGate, a standard ASCII transfer will occur. **Note that you can also use the 'File/Download Hex File' menu item to read a file.**

The file formats accepted by the SourceGate software are Motorola S-Record, Intel Hex, Texhex, HMI Binary, Extended Tek Hex, Microtec Research Absolute, Common Object File Format (COFF), Archimedes Debug (IAR), and Hitachi.

Once the Read Command has been executed, a File Load Status window will appear which will show the status of the read operation. The read operation can be aborted by pressing the **Cancel** button located in the File Load Status window.

When reading a file from the RS232 port, the format is:

RD([SUPDT])(Offset),Filename,(Flags)

When reading a file from the parallel port, the format is:

RDP([SUPDT])(Offset),Filename,(Flags)

Where:

SUPDT: are the available overrides

Offset: is a hexadecimal value representing the address offset

Filename: is a file name, valid to the host computer, that designates the file to be downloaded to the emulator. If no extension is specified, the default .ABS will be used.

Flags: are options to be used for each read. The options are defined as follows:

- A: Download hex files to the emulator using ASCII file transfer
- B: Download hex files to the emulator using binary file transfer
- C: Concatenate any symbols found reading a hex file with those which already exist in the internal symbol table (i.e. do not clear the symbol table on a read command)
- D: Use only the digits presented when looking up a symbol by value for display (Opposite of "Z")
- H: Upload/Download only the hex data records present in a hex file. Ignore all symbol records.
- S: Upload/Download only the symbol records of a hex file. Ignore all hex data records.
- M: Toggle between reading "S" record files as Motorola and Hitachi formats. Default is Motorola for Motorola microprocessors.
- Z: Pad a number with leading zeros when doing a lookup by value for symbol display (Opposite of "D").

The default option control flag settings are "BZ".

The 'RD' command opens the named format file (.ABS being the default extension) and reads its symbols into the internal SourceGate symbol table, and its hex data into the emulator's (or target system's) memory.

If the optional read bias is given, this offset value will be added to the addresses of all bytes read from the file, before these bytes are stored in memory. This offset is a hexadecimal number. If the filename is not given, the 'RD' command will use the filename given in the last 'RD' command.

Note: The default file extension used on read (load) and write (dump) operations can be changed by modifying the .SG2 file for your emulator. For read operations, the "DefaultLoadExtension" definition can be changed to reflect the desired extension. For write operations, change "DefaultDumpExtension".

RD(%) (Read Configuration File) Command

This command will read a *Lite* configuration file that was previously saved to a file with the W(%) command. When this file is read, the *Lite* will configure itself to the configuration specified in the file. This configuration file contains the information shown in the Configuration and Event windows and should be generated by the user after the Configuration and Event windows are defined. The user can then read this file as part of the start-up batch file or multiple users can define their own configuration files to use.

The format of the Read Configuration File command is:

RD(%) filename

Where:

filename: is the name of the configuration file to be read.

RD(!) (Read Firmware File Into Flash ROM) Command

This command allows the user to upload the latest firmware into the *Lite*'s Flash ROM. After this command is executed, the *Lite* will then operate from the version of the firmware that was read.

The format of this command is:

RD(!) filename

Where:

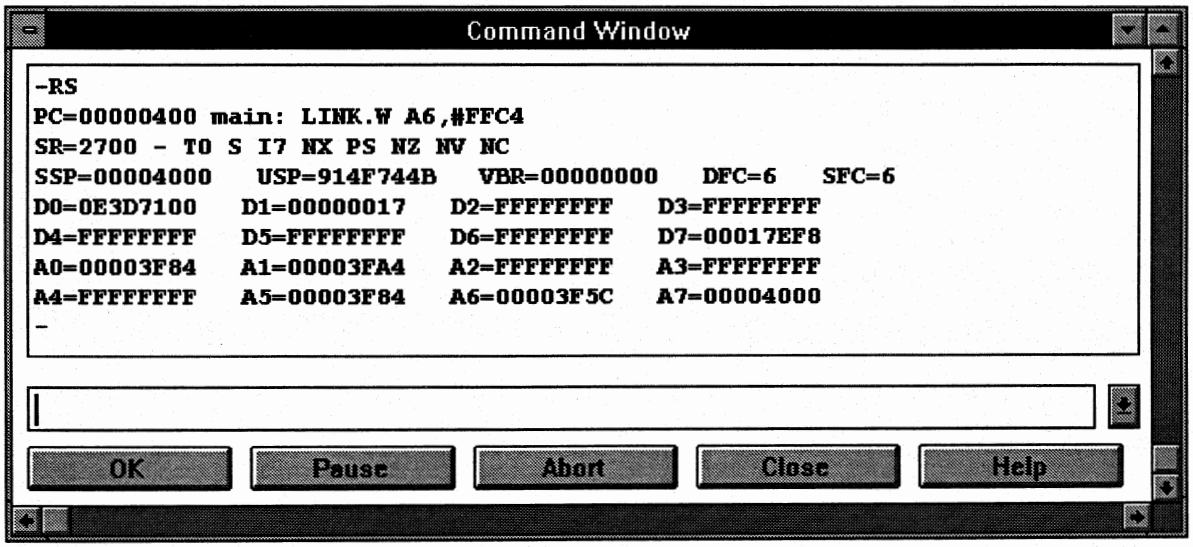
filename: is the name of the firmware file to be read into the flash ROM.

RP (Reset Processor, Continue Emulation) Command

The RP command allows you to reset the processor without stopping emulation.

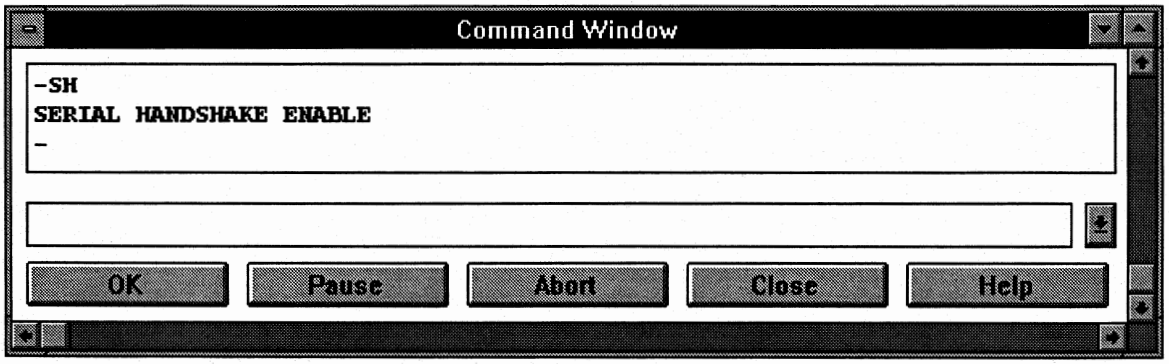
Note: The RP command can only be executed while emulation is in progress.

RS (Reset Processor, Stop Emulation) Command



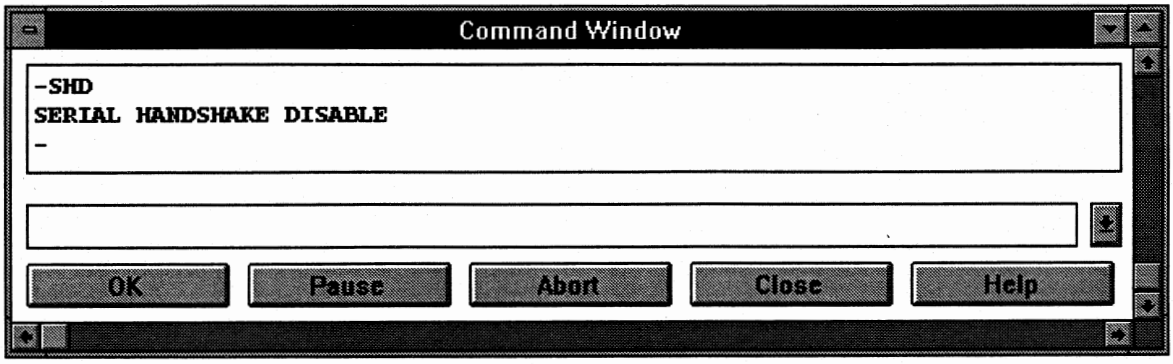
This command will issue a reset to the *Lite* processor. All registers will be updated to the value which they contain upon a processor reset. **Note that you can also use the 'Source/Reset Emulator' menu item and the "Run Status" window to reset the processor.**

SH (Show Status For SHD/SHE) Command



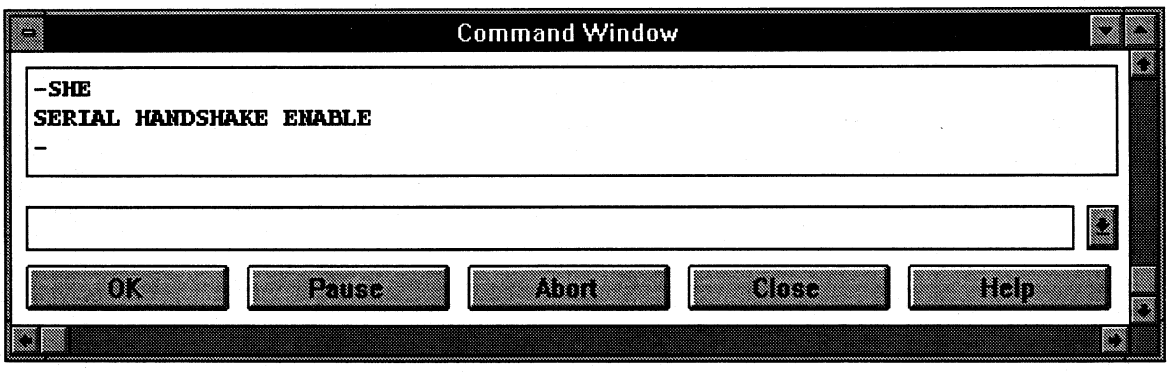
This command will show the status of the hardware handshake of the serial link between the emulator and the computer.

SHD (Serial Handshake Disabled) Command



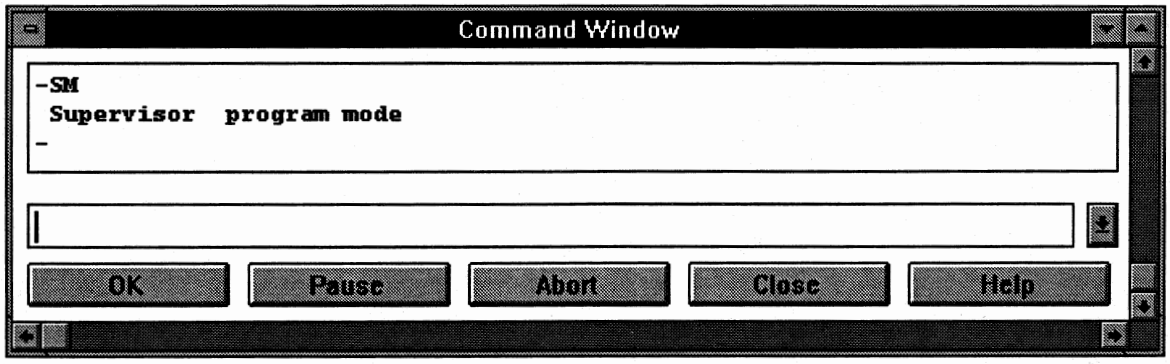
This command will disable the serial hardware handshake (RTS line) between the emulator and the computer. Using the SHE command or invoking SourceGate with the "-RTS" switch (which is the SourceGate default) will enable the hardware handshaking. This hardware handshaking may be needed to operate the emulator at the higher baud rates.

SHE (Serial Handshake Enabled) Command



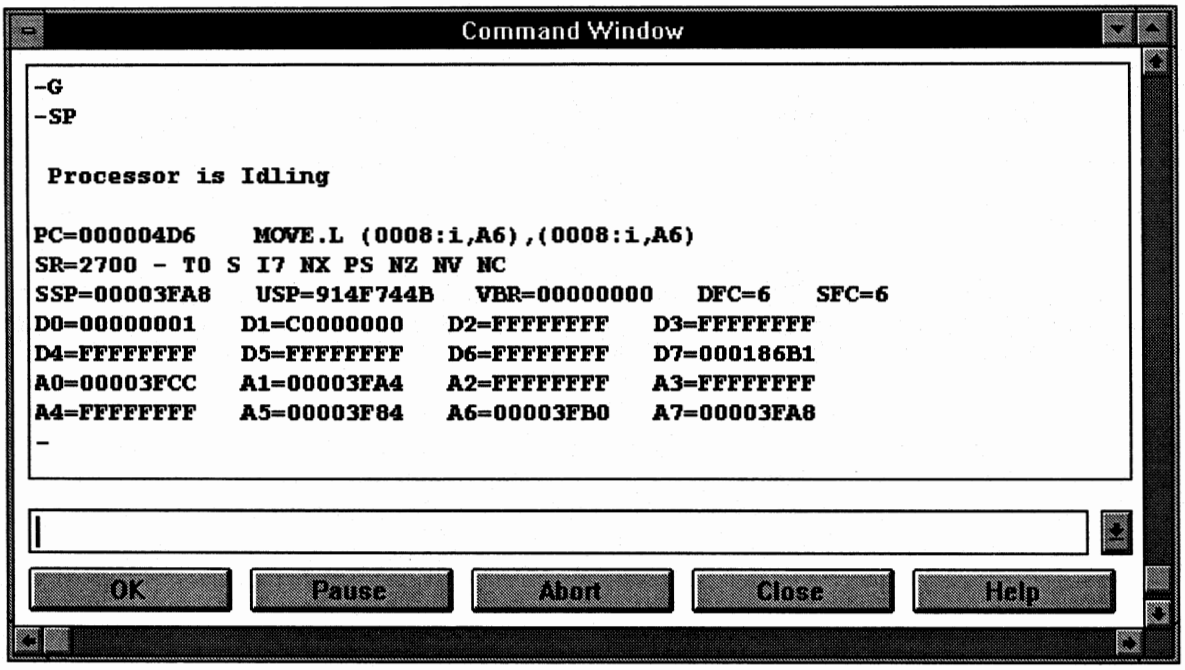
This command will enable the serial hardware handshake (RTS line) between the emulator and the computer. Using the SHD command or invoking SourceGate with the "-NORTS" switch will disable the hardware handshaking. This hardware handshaking may be needed to operate the emulator at the higher baud rates.

SM (Supervisor Memory Mode) Command



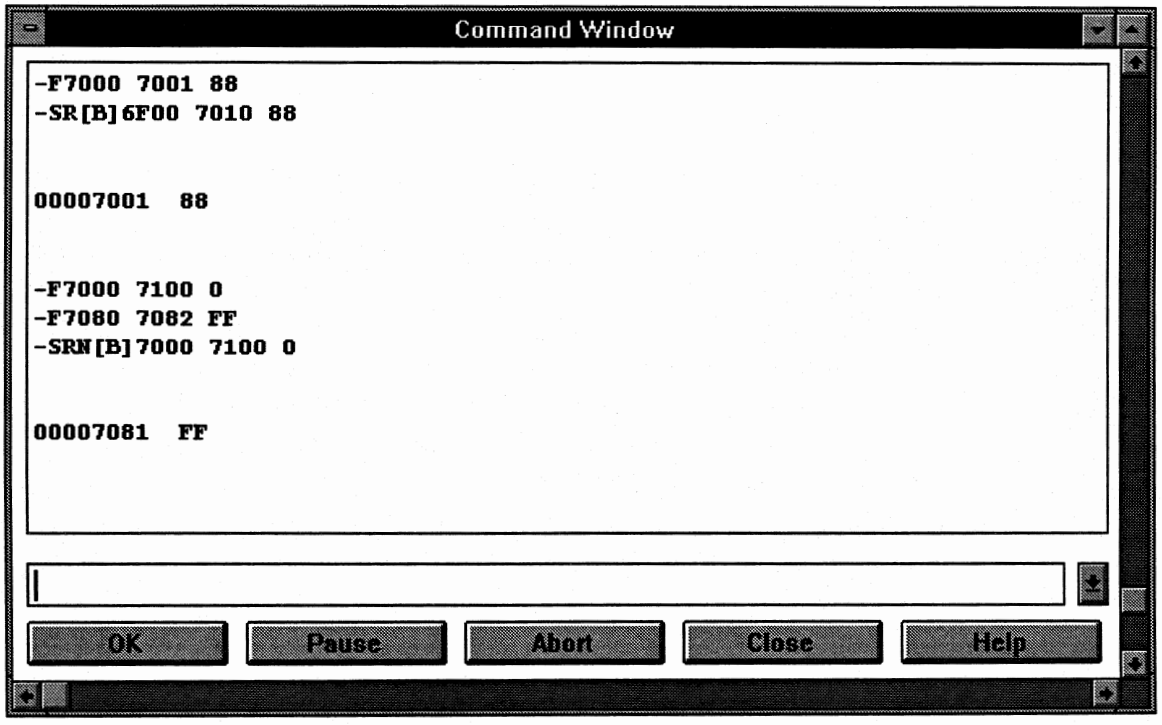
The execution of this command sets the default memory category to be Supervisor Memory mode. Using the PM (Program Mode) or DM (Data Mode) commands further define the default memory mode.

SP (Stop Emulation) Command



This command is used to stop emulation in order to regain control of the processor. Upon entering this command, emulation will break, and the message "PROCESSOR IS IDLING" will be displayed and the register window will be updated with the current register values. **Note that you can also use the 'Source/Stop' menu item and the "Run Status" window to stop emulation.**

SR/SRN (Search Block Of Memory) Command



This command can be used to search memory between two addresses for a particular data pattern. When a match is found, the address is displayed. To search for the next occurrence, hit the Carriage Return key. To terminate this command, type a period. If no match is found, then nothing is displayed. **Note that you can also use the Memory window to search the contents of memory.**

There is also a Search Not option available. In this mode the value is searched for, but the command displays the address of all memory locations in the given range that do not match the search value. This is useful when checking a block of memory that should contain a constant value, such as zero or the value of a previous Fill command.

The format for the Search command is:

SR([SUPDLWBTR]) Baddr Eaddr Val

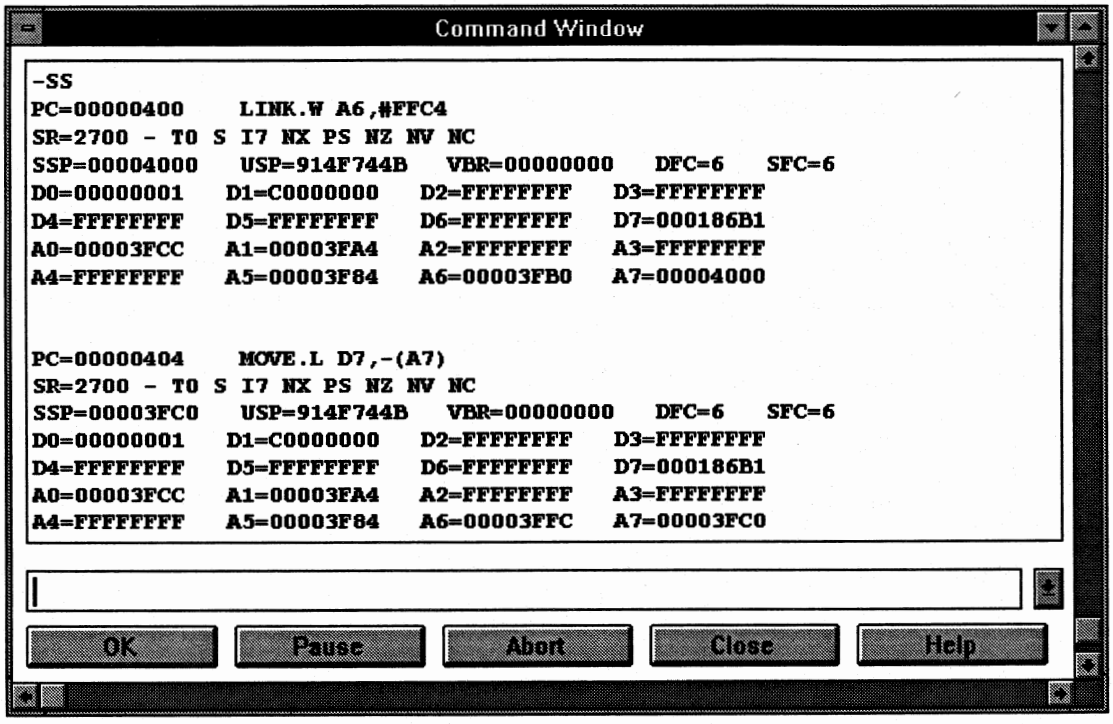
The format for the Search Not command is:

SRN([SUPDLWBTR]) Baddr Eaddr Sval

where:

Baddr:	is the hexadecimal beginning address of the block of memory to be searched
Eaddr:	is the hexadecimal ending address of the block of memory to be searched
Val:	is the data value to be searched for and will be 1 byte or word value or a 1-20 character ASCII string enclosed in single quotes.
Sval:	is the data value to be searched for and will be a byte value, a word value or one or two ASCII characters enclosed in quotes.
SUPDLWBTR:	are the available overrides

SS (Single Step) Command



This command is used to execute one instruction at a time. Each time a step is performed, the address and instruction to be executed are displayed along with the register values from the previous instruction executed. Therefore, multiple steps will display a history of program flow. There are four options for operating the Single Step Command. The first option is to single step every time a carriage return is entered. To terminate this mode, type a period. The second option is to single step a specified number of times. The third option is to single step continuously. The fourth option allows you to single step "Until" some condition. To terminate the second, third, and fourth options, a carriage return should be entered.

The format of the single step command is as follows:

SS[<count>][X] [Until clause] [Q/Q1/Q2] [Skip]

Where:

SS:	Steps once each time a <cr> is entered. A period (.) aborts stepping
SS<count>:	Steps <count> times without waiting for a <cr>. A <cr> aborts stepping
SSX:	Steps continuously until a <cr> is entered
SS Until clause:	Steps until the condition in the clause is true. A <cr> aborts stepping

The "Quiet" option is invoked by appending a "Q" to the end of the command. In this mode, only the first and last step are displayed (e.g. SSX Q will step automatically until a <cr> is entered showing the first and last instructions stepped). If "Q1" is used, only the last step is shown. If "Q2" is used, all steps are shown in the form "Addr Instruction" where "Addr" is the value of the PC and "Instruction" is the assembly language instruction pointed to by that PC value. In this mode, all register values except the PC are suppressed.

The optional "SKIP" parameter causes all subroutine calls to be shown as a single step. The subroutine called will be executed but the information shown while stepping will not include the instructions in the subroutine, only the call to it. If the subroutine called, in turn, calls other subroutines, or if the subroutine called contains a large number of instructions, there may be a noticeable wait until the program returns to the main program.

The Until clause is defined as follows:

```
OPERAND1 OPERATION OPERAND2

Where OPERAND1 can be:

Reg <reg> - Register name
Long at <addr> - Long word value at address <addr>
Word at <addr> - Word value at address <addr>
Byte at <addr> - Byte value at address <addr>

OPERAND2 can be:

Reg <reg> - Register name
#<value> - Immediate value (size determined by OPERAND 1)
<addr> - Value at <addr> (size determined by OPERAND 1)

OPERATION can be:

 = - OPERAND1 equal to OPERAND2
== - OPERAND1 equal to OPERAND2
!= - OPERAND1 not equal to OPERAND2
<> - OPERAND1 not equal to OPERAND2
>< - OPERAND1 not equal to OPERAND2
 > - OPERAND1 greater than OPERAND2
 < - OPERAND1 less than OPERAND2
>= - OPERAND1 greater than or equal to OPERAND2
=> - OPERAND1 greater than or equal to OPERAND2
<= - OPERAND1 less than or equal to OPERAND2
=< - OPERAND1 less than or equal to OPERAND2
```

Example:

SS Until word at 5000 = #1234

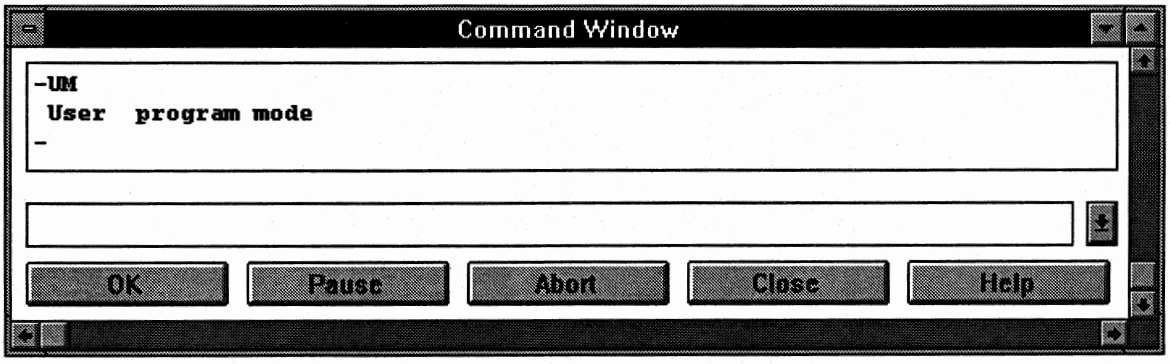
Would stop stepping when 1234H appears at address 5000H

It should be noted that this command takes into account the way data is stored in memory based on the microprocessor manufacturer. If, for example, you wished to stop on a word value of 1234H being in location 5000H, the emulator will look for a 1234H at location 5000H for Motorola processors and 3412H at location 5000H for Intel, Hitachi, and Zilog processors.

ST (Show Emulator Status) Command

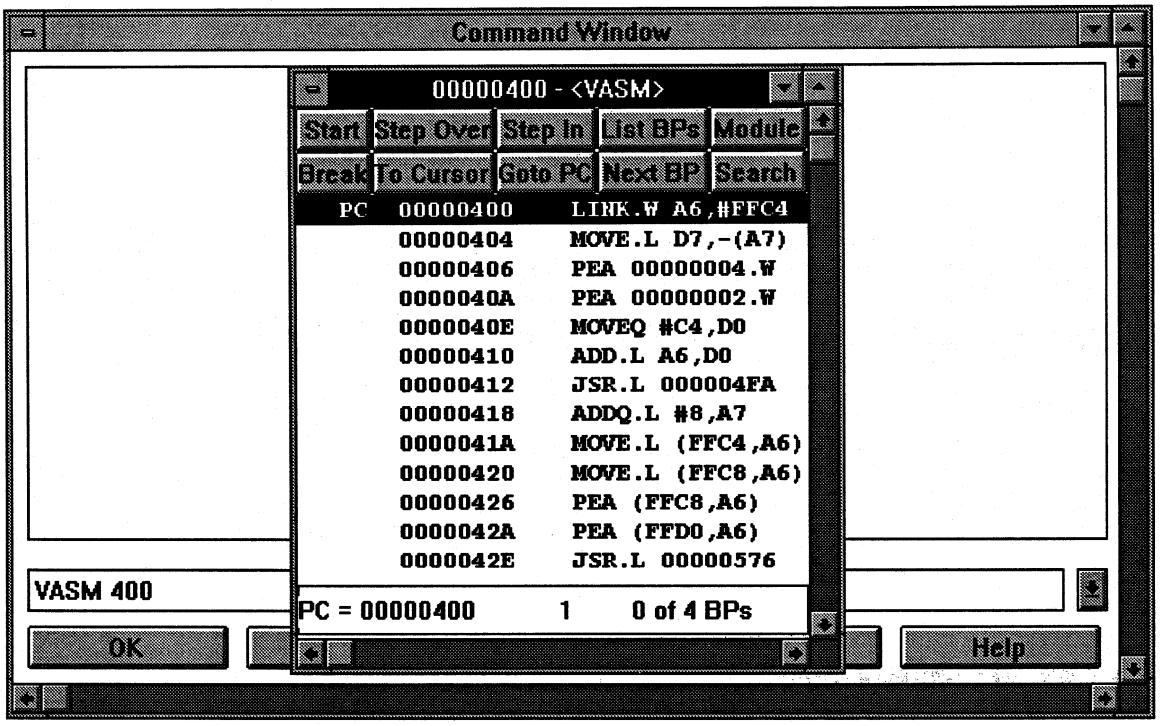
Executing the Status command will display the current status of the emulator.

UM (User Memory Mode) Command



The execution of this command sets the default memory category to be User Memory mode. Using the PM (Program Mode) or DM (Data Mode) commands further define the default memory mode.

VASM (View Assembly) Command



This command will open an assembly Code window beginning at the specified address. **Note that you can also use the 'Source/VASM' menu item to open a VASM window.**

The format of the VASM command is:

VASM <addr>

Where:

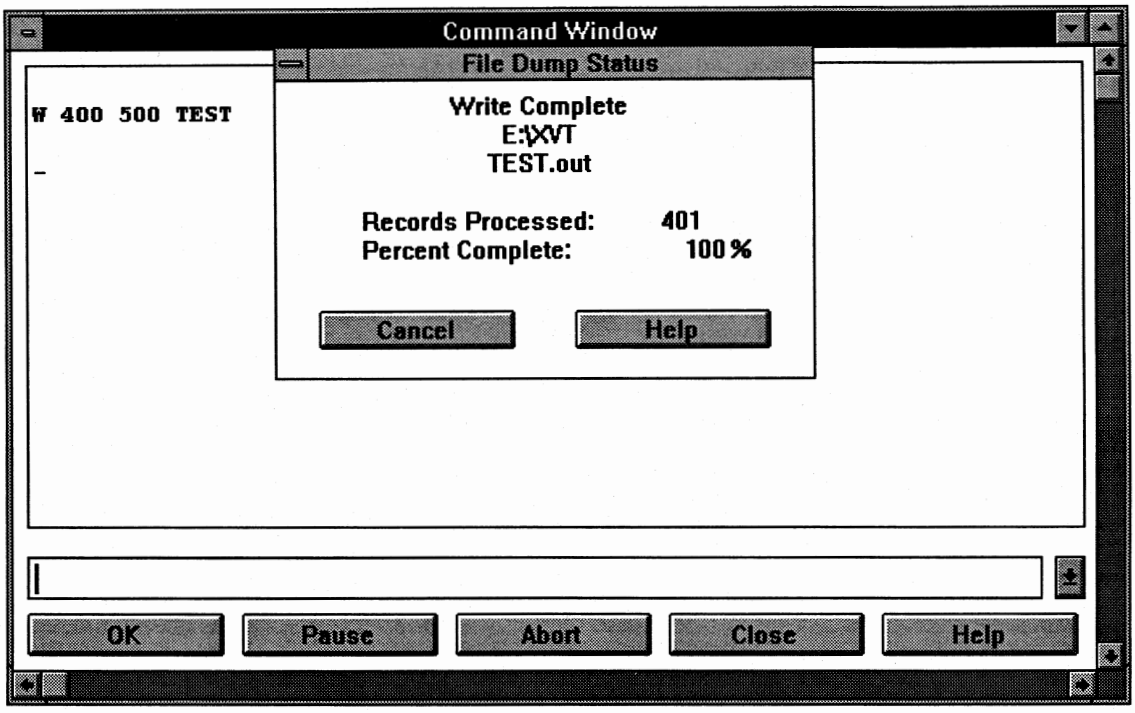
addr: is the beginning address of the VASM window

If no address is entered, the VASM window will begin with the address of the current program counter.

VER (Display Firmware Version) Command

This command displays the current version and release date of the emulator firmware.

W (Write Memory Contents To File) Command



This command is used to transfer files from the emulation memory to a host computer. The size of the block being transferred is specified by the two address fields activated by this command. When using the Huntsville Microsystems, Inc. communication software (SourceGate), the block will be written as a Motorola S-Record file with symbol information added to the beginning of the file. This symbol information is in a format that is easily read by HMI software. The extent of this file defaults to .HEX. This symbol/hex file is written to disk on the host computer under the specified filename. **Note that you can also use the 'File/Upload Hex File' menu item to read a file.**

Once the Write Command has been executed, a File Load Status window will appear which will show the status of the write operation. The write operation can be aborted by pressing the **Cancel** button located in the File Load Status window.

The format for writing to the RS232 port is:

W([SUPDT]) Baddr Eaddr Filename (Flags)

Where:

SUPDT:	are the available overrides
Baddr:	is the hexadecimal beginning address of code to be written
Eaddr:	is the hexadecimal ending address of code to be written
Filename:	is a filename, valid to the host computer, that designates the file to be uploaded (written) to the host computer. If no extension is specified, the default .HEX will be used.
Flags:	are options to be used for each write The options are defined as follows:
H:	Upload/Download only the hex data records present in a hex file. Ignore all symbol records.
S:	Upload/Download only the symbol records of a hex file. Ignore all hex data records.

The 'W' command opens the given file for writing, and writes the contents of the SourceGate internal symbol table into this file, along with the data bytes in memory, from Baddr through Eaddr. 'Baddr' and 'Eaddr' are hexadecimal numbers (or symbols).

Note: The default file extension used on read (load) and write (dump) operations can be changed by modifying the .SG2 file for your emulator. For read operations, the "DefaultLoadExtension" definition can be changed to reflect the desired extension. For write operations, change "DefaultDumpExtension".

W(%) (Write Configuration File) Command

This command will write the *Lite's* configuration information to a file. This configuration information is that information displayed in the Configuration and Event windows. When this file is read using the RD(%) command, the *Lite* will set itself up to reflect the configuration defined in this file. This command should be used to save the configuration of the *Lite* after the Configuration and Event windows have been defined.. This allows multiple users to set-up the *Lite* for their particular application by simply writing and then reading their own configuration information.

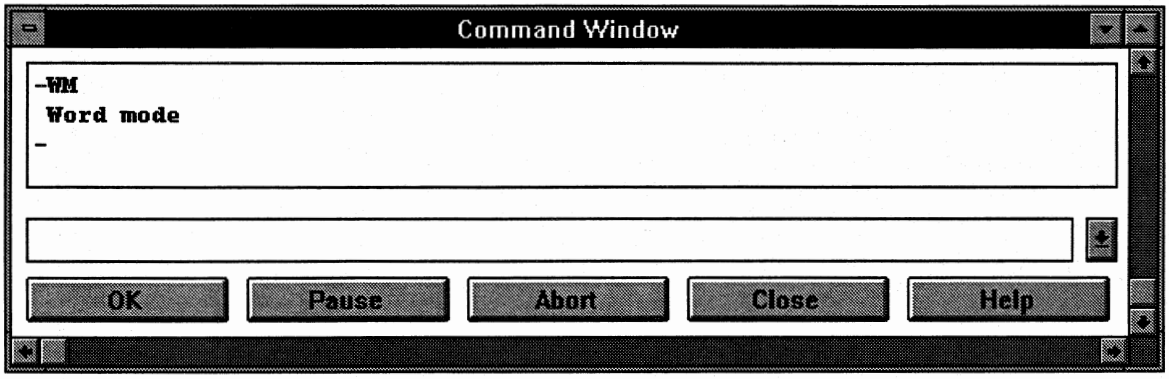
The format of the Write Configuration File command is:

W(%) filename

Where:

filename: is the name of the configuration file to be written.

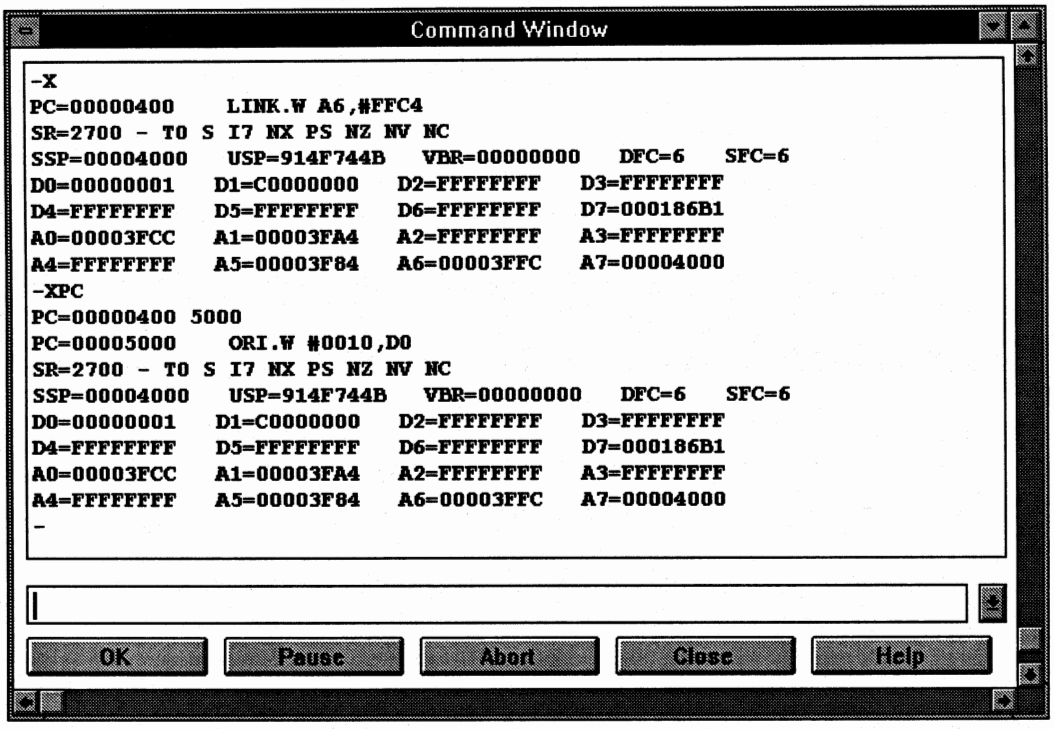
WM (Word Display Mode) Command



This command sets the default for input or output values to "Word Mode". While in Word mode, commands such as D (Dump memory) will display values in Word mode instead of long word mode or byte mode, which are the other choices (see LM and BM).

In addition, commands which require an input value will be affected by this command. For example, if the Fill command, F 8000 9000 33 is executed while in Word mode, each word of memory from 8000H through 9000H will contain 0033H. If this same command was executed with the emulator in long word mode, then 00000033H would be stored in each long word location from 8000H to 9000H. If this same command was executed with the emulator in byte mode, then 33H would be stored in each byte location from 8000H to 9000H.

X (Examine/Change Registers) Command



This command allows you to examine and modify the registers and flags (status register). There are three options to the examine command. The first option is to display all registers and flags. The second option is to display a specific register and allow it to be modified. The last option is to allow the flags to be modified by typing in the condition code status. **Note that you can also use the Register window to display and change the register contents.**

The format for displaying the registers and condition codes is:

X

The format for displaying a specific register and allowing it to be modified is:

XReg

The format for displaying just the condition code values is:

XFL

Where:

Reg: is any processor register such as A1 or SR

When changing a register using the "Xreg" command, you will be shown the current value of the register. To change it, simply type in the new value and then press the "ENTER" key. The register will then contain the new value.

When changing a condition code value using the "XFL" command, you will be shown the current value of the status register and corresponding condition codes. To change condition codes, enter the codes to be changed and then press the "ENTER" key. The register display will then show the modified condition codes.

An explanation of each of the condition codes is listed below:

SR Bit	Flag or Condition Code Name	Not Active Code	Active Code
15	Trace Enable T1	T0 or T1	T2 or T3
14	Trace Enable T0	T0 or T2	T1 or T3
13	Super/User State	U	S
12	Mast/Int State	I	M
11	Not Defined	--	--
10	Int Priority Mask 2		
9	Int Priority Mask 1	I0 through I7	
8	Int Priority Mask 0		
7	Not Defined	--	--
6	Not Defined	--	--
5	Not Defined	--	--
4	Extend	NX	XT
3	Negative	PS	NG
2	Zero	NZ	ZR
1	Overflow	NV	OV
0	Carry	NC	CY

Use these codes, after entering the "XFL" command, to change to a different condition code. For example, if you perform an "XFL" command and are shown the following:

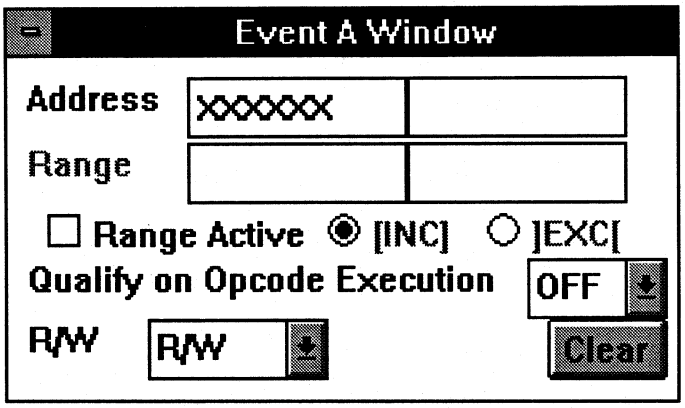
SR = 2709 NT S I7 NX NG NZ NV CY

then you enter "U I6 ZR" and press the "ENTER" key, you will see the following:

SR = 060D NT U I6 NX NG ZR NV CY

Setting Up Events

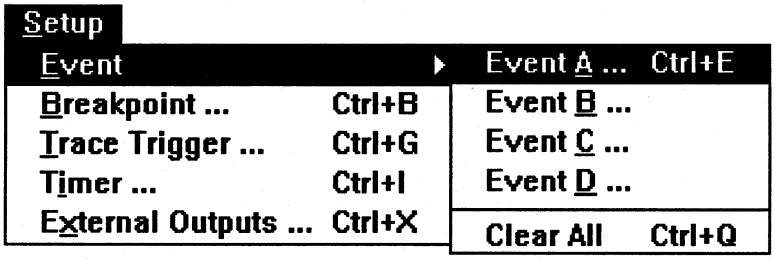
Events are used to define conditions for breakpoints, trigger points, interval timer start and stop conditions, and external output signal conditions. There are four event windows: A, B, C, and D. Each of these events are independent and can be setup for specific conditions. A typical Event window has the following format:



Note that 68340 will have a 32-bit address space verses 24-bit for the 68331/68332/68HC16. An overview of the operation of the Event windows follows:An overview of the operation of the Event window follows:

Opening The Event Windows

To open the Event windows, choose 'Setup/Event'. Click on 'Event' again to see a listing of the four available events:



Click on the desired event in order to open that event window. An alternate method of opening the Event A window is to use the accelerator key combination Ctrl+E. To clear all events back to their default values, use Ctrl+Q.

Event Window Fields

The Event windows have several fields that can be defined as part of the event condition. An overview of all possible event fields for all *Lite* emulators is given in the sections that follow.

Event Address Field

This field allows you to enter a hexadecimal address for the event by moving the insert pointer to the desired location and pressing the left mouse button. Binary values can be entered by clicking on the 'Address' field and entering the desired values in the Event Address Binary window. Valid values for this field are 0-F and 'X' which is the don't care condition.

Event Address Range Field

If address ranges are enabled by choosing "Range Active", this field becomes the hexadecimal ending address of the range. You can enter a hexadecimal ending address for the range by moving the insert pointer to the desired location and pressing the left mouse button. If ranges are disabled, the "Range" label will be shaded. The binary representation of the ending address can be shown and changed by pressing the left mouse button on the "Range" label. The field in the right-most column is the Event Symbol entry field where symbols can be entered. Valid values for this field are 0-F and 'X' which is the don't care condition.

Event Symbol Entry Field

The event symbol entry fields are located in the far right columns of the address, range, and data fields. Symbols can be entered by first moving the insert pointer to this field and pressing the left mouse button followed by the right mouse button. You can then choose the symbol to be entered. The value for this symbol is then placed in the hexadecimal field.

Event Range Active Field

This field allows you to define if a particular range is active or inactive. The status of this field will toggle by moving the cursor to the field and pressing the left mouse button.

If the range is made active, you can define if the range is inclusive (within the range) or exclusive (outside the range) by choosing either "[INC]" or "]EXC[", respectively, and pressing the left mouse button.

Event 'Qualify On Opcode Execution' Field

An additional field can be defined to allow the event to be qualified on an opcode execution. Choosing "ON" will assure that the event will become valid when an opcode is executed verses an opcode fetch at the defined address or address range.

Event 'R/W' Field

You can define an event to become valid only when the Read/Write line is a certain state. This enables the defined event to be further qualified. The options for the Read/Write condition as as follows:

Write:
Event will become true when the defined address condition becomes valid and the Read/Write line is active low (Write) showing that a write operation is being performed at the defined address/address range.

Read:
Event will become true when the defined address condition becomes valid and the Read/Write line is active high (Read) showing that a read operation is being performed at the defined address/address range.

R/W:
Event will become true when the defined address condition becomes valid regardless of the state of the read/write line.

Event Window Buttons

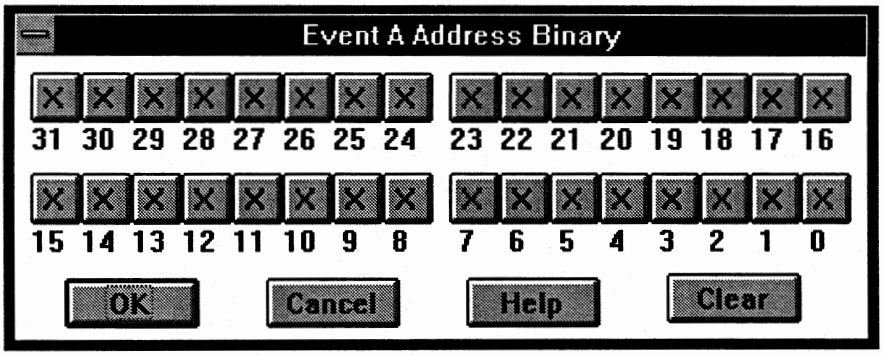
The following button is available in the Event windows.

Clear
The **Clear** button will set all event fields back to their default values. All hexadecimal fields will contain 'X's (don't cares) and ranges will be disabled.

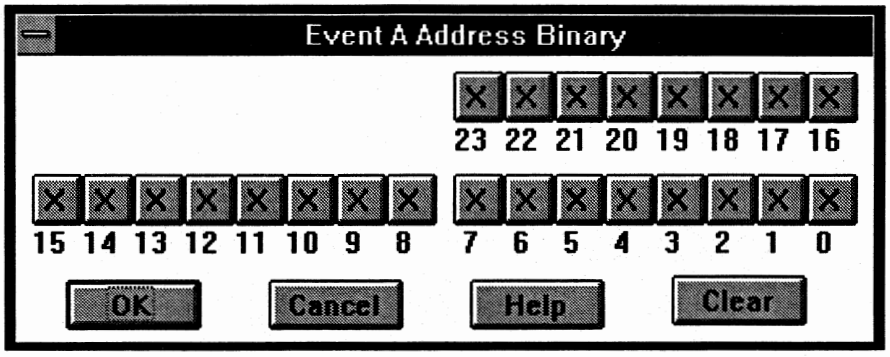
Scroll Box
The **Scroll Box** buttons will allow you to choose from a pre-defined list of options.

Event Binary Address Window

If you click on the 'Address' label in the Event window, you will open an 'Event Address Binary' window for that event. This window will look as follows for the 68340:



and will look like this for the 68331/68332/68HC16:



The 'Event Address Binary' window allows you to define an address in binary instead of hexadecimal. To define the event to be valid when an address bit has a particular level on it, place the mouse cursor on the button for the desired address bit and then press the left mouse button. Each press of the left mouse button will toggle through the available choices for the address bit button. You can choose between a "1" for a logic high value, a "0" for a logic low value, and "X" for "Don't Care" which will disable that address bit from influencing the event.

Event Binary Address Window Buttons

The following buttons are available in the Event Binary Address window:

0,1,X

You can toggle the bit value for each address bit by moving the mouse pointer to the desired address bit and pressing the left mouse button. This will toggle through all possible choices for the address bit (0, 1, and X (Don't Care)).

OK

The **OK** button will accept the values set in the Event Binary Address window.

Cancel

The **Cancel** button will ignore any changes made in the Event Binary Address window and will close the window.

Help

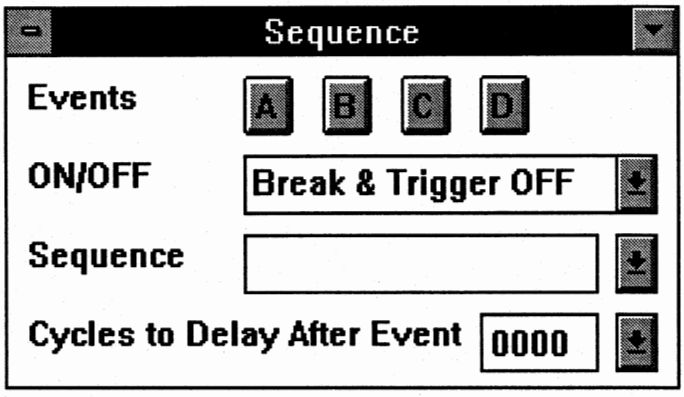
The **Help** button will invoke help for this window.

Clear

The **Clear** button will clear all binary address bit values back to the default value of "X" which is the Don't Care condition.

Defining Breakpoints and Trace Triggers

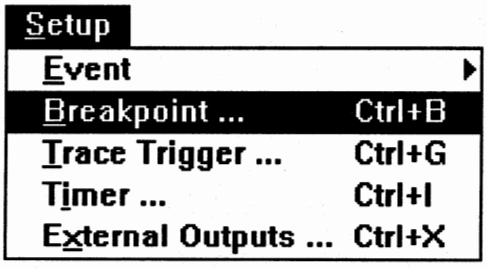
Breakpoints are used to stop emulation when a certain condition or set of conditions occurs. The Trace Trigger, as it's name suggests, is used to trigger the trace buffer when a certain condition or set of conditions occurs. These conditions are defined by the Event windows and are then sequenced together, using the Sequence window, to create the breakpoint or trace trigger condition. The Sequence window has the following format:



An overview of the operation of the Sequence window follows:

Opening The Sequence Window

To open the Sequence window, choose 'Setup/Breakpoint' or 'Setup/Trace Trigger':



An alternate method of opening the Sequence window is to use the accelerator key combination Ctrl+B or Ctrl+G.

Sequence Window Fields

The Sequence window has several fields that determine how breakpoints and trace triggers occur. An overview of these fields is given in the sections that follow.

Sequence Window 'ON/OFF' Field

There are three choices for the 'ON/OFF' field. These choices are listed as follows:

Break & Trigger OFF:
The definded events will not be used to break emulation or trigger the trace buffer.

Break ON:
When the defined break sequence is reached, the Lite will continue to run until it executes the number of cycles specified in the "Delay after Event" field. Emulation will then stop.

Trigger ON:
When the defined trigger sequence is reached, the *Lite* will continue to record data in the Trigger Trace buffer until the processor runs the number of cycles specified in the "Delay after Event" field. At that point, the Trigger buffer will be held for future reference and the *Lite* will continue to run without interruption.

Sequence Window 'Sequence' Field

Once the desired events have been defined, they can be sequenced together to created a condition to break emulation or trigger the trace. This ability to sequence the events together gives the *Lite* the power to break or trigger on complex conditions.

Sequence Window 'Cycles to Delay After Event' Field

The "Cycles to Delay After Event" field allows you to determine how many cycles are executed after the Break condition is reached (if "Break ON" is the activator) or how many cycles of information get recorded in the Trigger buffer after the trigger point is reached (if "Trigger ON" is the activator). This delay value can be manipulated to display information around a desired point of interest. For example, if the activator is "Trigger ON" and the delay value is 800H, then the trigger point will be located in the middle of the trigger buffer. Since 800H (2K) is specified as the delay value (how much information gets recorded in the trace after the trigger point), and the buffer is 4K deep, the user will see up to 2K of information before the trigger point, the trigger point, and then 2K (800H) cycles of information after the trigger point. If 0 was specified as the delay value, then the trigger point will be shown as the last cycle in the trace buffer since the delay is specified to be 0.

Sequence Window Buttons

The following buttons are available in the Sequence window:

A,B,C,D

The **A,B,C,D** buttons allow you to activate the desired event window without leaving the Sequence window. This simplifies setup.

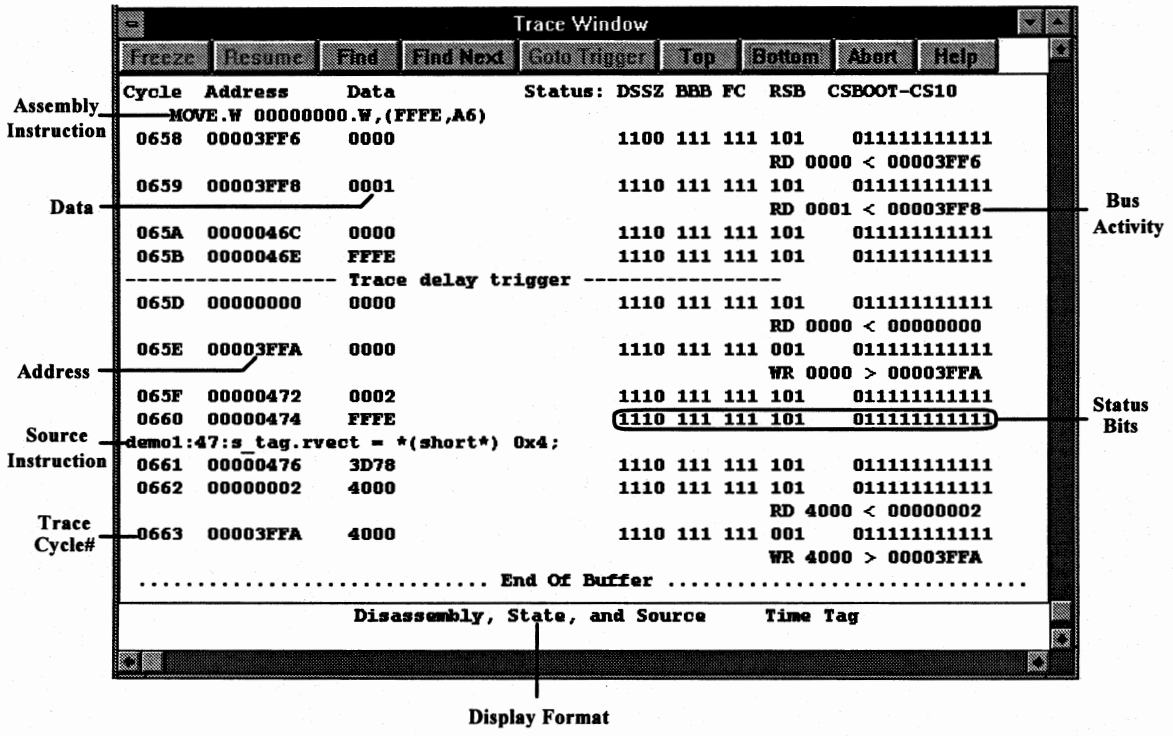
Scroll Box

The **Scroll Box** button will allow the user to scroll through the pre-defined sequences, ON/OFF selections, and any previously defined delay cycles.

Working With The Trace Window

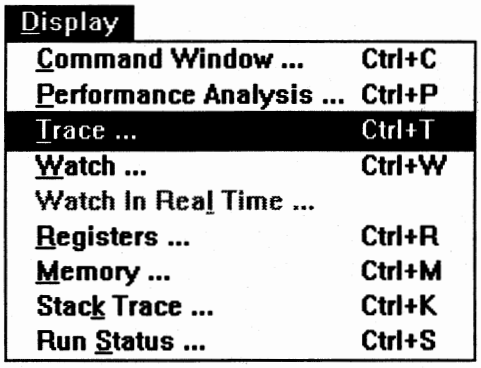
The Trace window allows the user to view and search through the trace buffer and to initiate and view the Freeze trace. The trace buffer can be displayed in a combination of disassembly, state, and source. Other functions of the Trace window include trace search and trace print.

The Trace window is updated each time emulation is terminated through a breakpoint, stop command, reset command, or when a freeze trace command is issued. The information contained in the Trace window is dependent on the processor being emulated. An example Trace window is shown below:



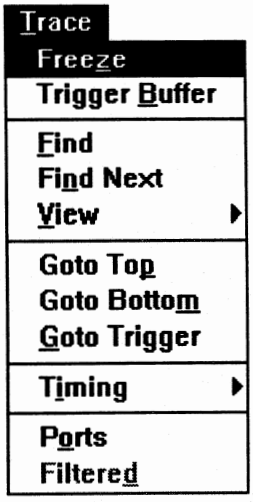
Opening The Trace Window

To open the Trace window, choose 'Display/Trace':



An alternate method of opening the Trace window is to use the accelerator key combination Ctrl+T.

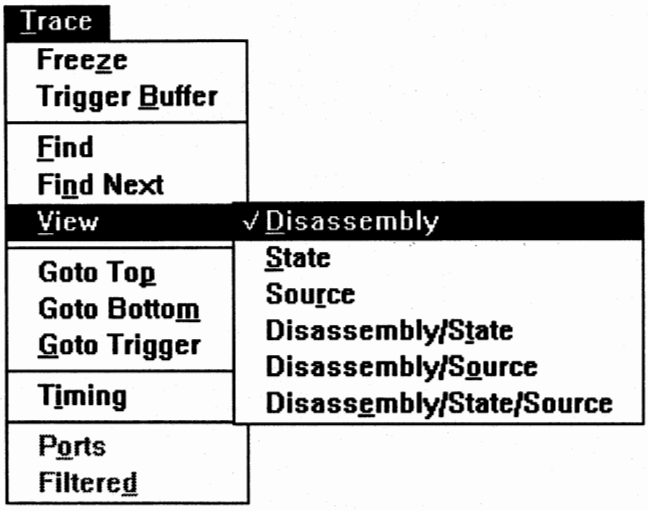
When the trace window is open, an additional menu item labeled "Trace" is shown in the SourceGate II menu bar. Selecting this menu item will show you an additional set of selections which allow you to define what information is shown in the trace buffers:



An overview of each of these menu items is given in the sections that follow.

Trace Display Modes

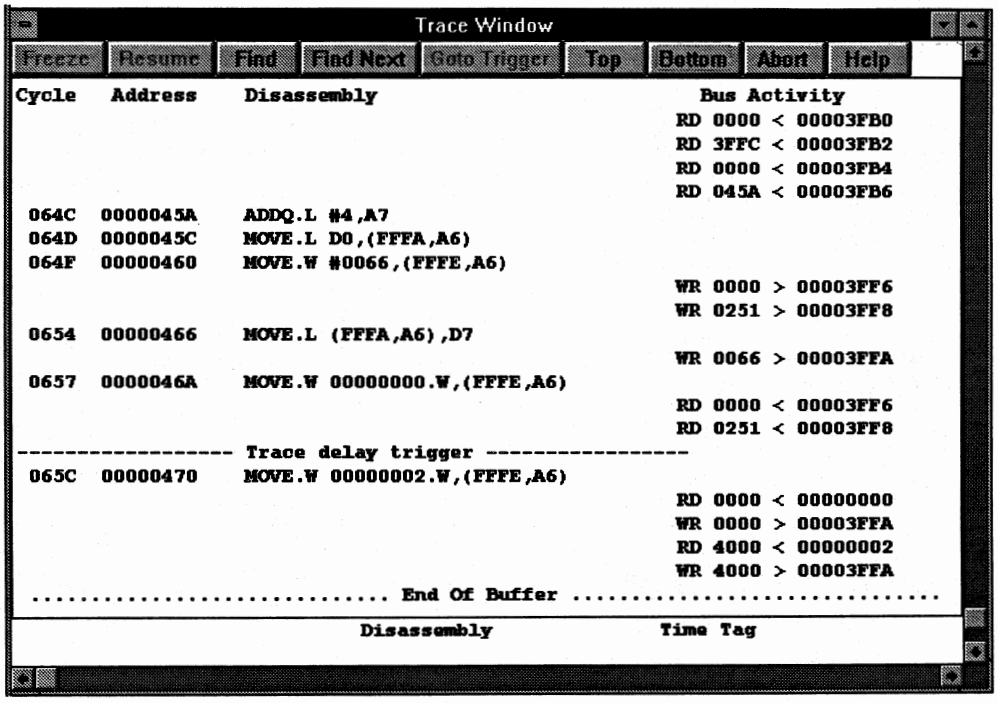
The display format of the trace display can be chosen by using one or a combination of the "View" options. The available options will be shown in a submenu with the following format:



An explanation of each of these options is given in the following sections.

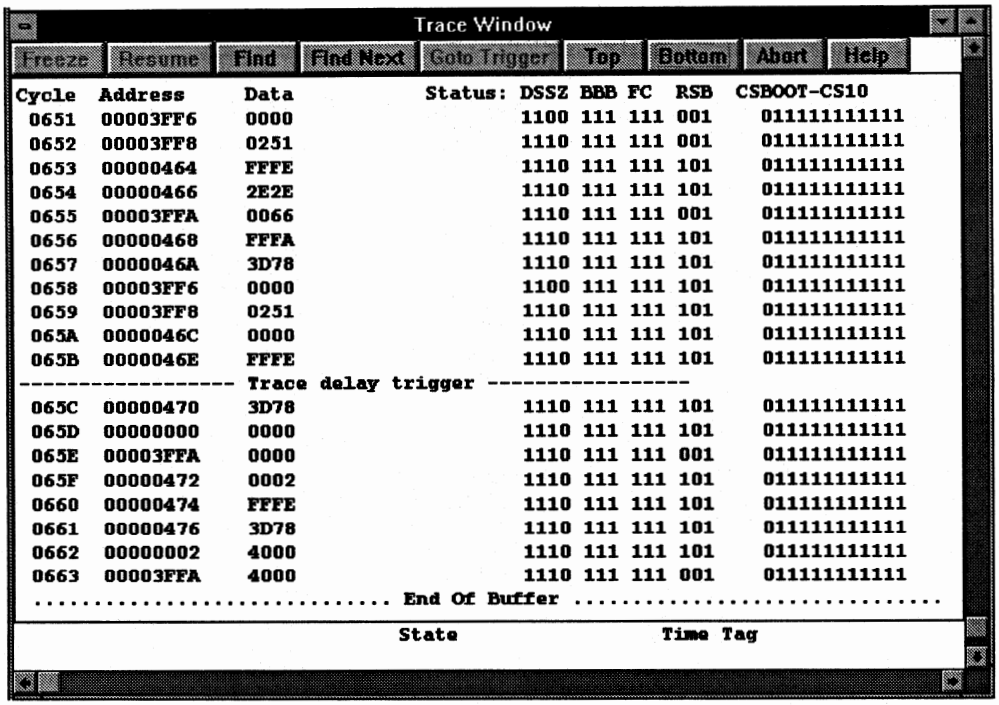
Disassembly

The trace buffer will show only disassembly information which includes cycle, address, disassembled instructions, and bus activity. All state and source code information will be suppressed. To select this mode, choose 'Trace/View/Disassembly'. An example *Lite* Trace window is shown below in 'Disassembly' format:



State

The trace buffer will show only state information which includes cycle, address, data, status bits, and bus activity. All disassembly and source code information will be suppressed. To select this mode, choose 'Trace/View/State'.



The status bits shown in the trace buffer are defined as follows when using a 68331, 68332, or 68HC16 *Lite* unit:

DSSZ	DSACK1'(PORTE1), DSACK0'(PORTE0), SIZ1(PORTE7), SIZ0(PORTE6)
BBB	BR'(CS0'), BG'(CS1'), BGACK'(CS2')
FC	FC2(CS5'), FC1(CS4'), FC0(CS3')
RSB	R/W', SHOW, BERR'
CSBOOT-CS0	CSBOOT', CS0'(BR'), CS1'(BG'), CS2'(BGACK'), CS3'(FC0), CS4'(FC1), CS5'(FC2), CS6'(A19), CS7'(A20), CS8'(A21), CS9'(A22), CS10'(A23)

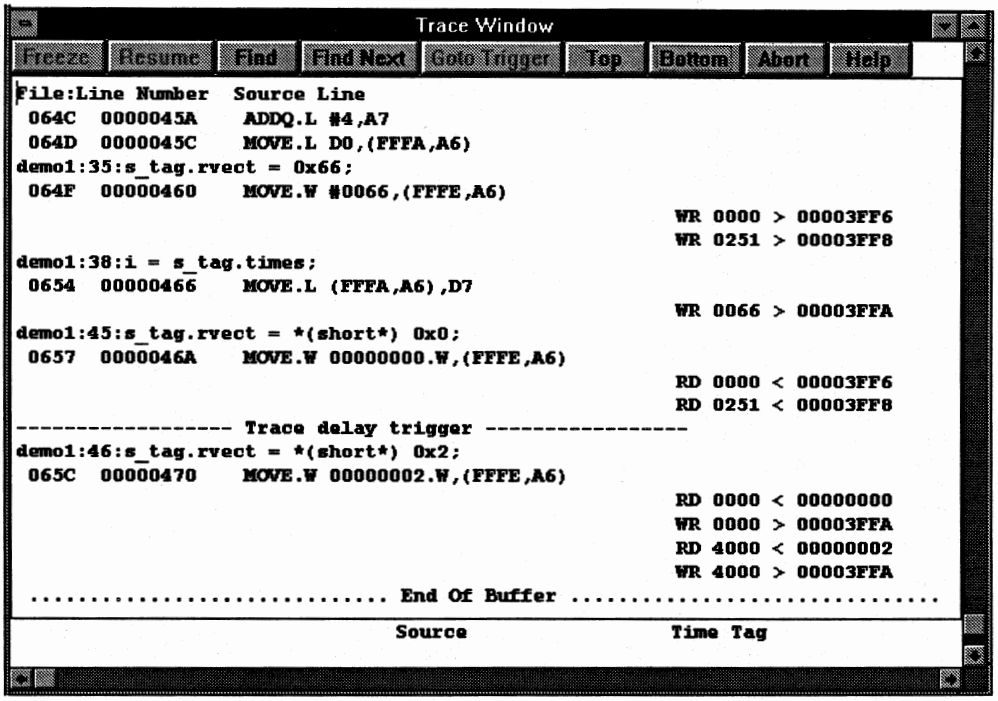
The status bits shown in the trace buffer are defined as follows when using a 68340 *Lite* unit:

DSSZ	DSACK1', DSACK0', SIZ1, SIZ0
FC	FC3, FC2, FC1, FC0
RSB	R/W', SHOW, BERR'
CS0-CS3	CS0'(AVEC'), CS1'(IRQ1'/PORTB1), CS2'(IRQ2'/PORTB2), CS3'(IRQ4'/PORTB4)

Note: () is alternate function depending on how pin is programmed

Source

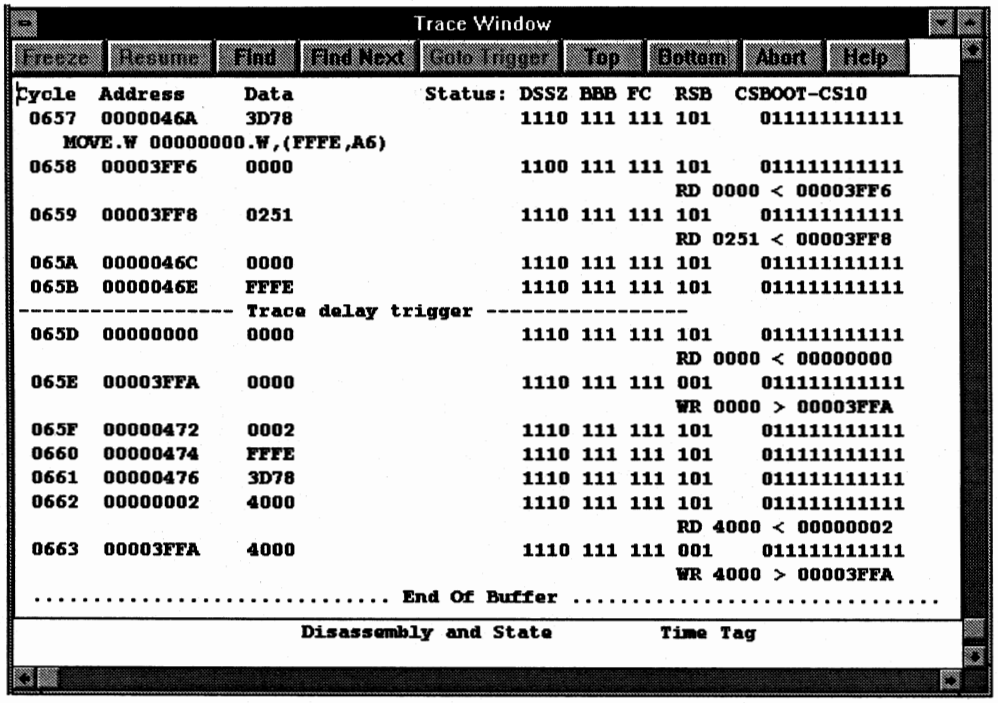
The trace buffers will show only high-level source information which includes the module name, line number of the module, and the actual high-level line of code. To select this mode, choose 'Trace/View/Source'.



It should be noted that some disassembled instructions and bus activity may also be shown while in this mode. This information usually appears when compiler routines are traced that were written in assembly language and, thus, have no high- level source code available for them. SourceGate has no choice but to display this information as the "source" associated with the displayed cycles.

Disassembly/State

The trace buffers will show both disassembly and state information which includes cycle, address, data, status bits, and bus activity. All high-level source information will be suppressed. To select this mode, choose 'Trace/View/Disassembly/State'.



The status bits shown in the trace buffer are defined as follows when using a 68331, 68332, or 68HC16 *Lite* unit:

DSSZ	DSACK1'(PORTE1), DSACK0'(PORTE0), SIZ1(PORTE7), SIZ0(PORTE6)
BBB	BR'(CS0'), BG'(CS1'), BGACK'(CS2')
FC	FC2(CS5'), FC1(CS4'), FC0(CS3')
RSB	R/W', SHOW, BERR'
CSBOOT-CS0	CSBOOT', CS0'(BR'), CS1'(BG'), CS2'(BGACK'), CS3'(FC0), CS4'(FC1), CS5'(FC2), CS6'(A19), CS7'(A20), CS8'(A21), CS9'(A22), CS10'(A23)

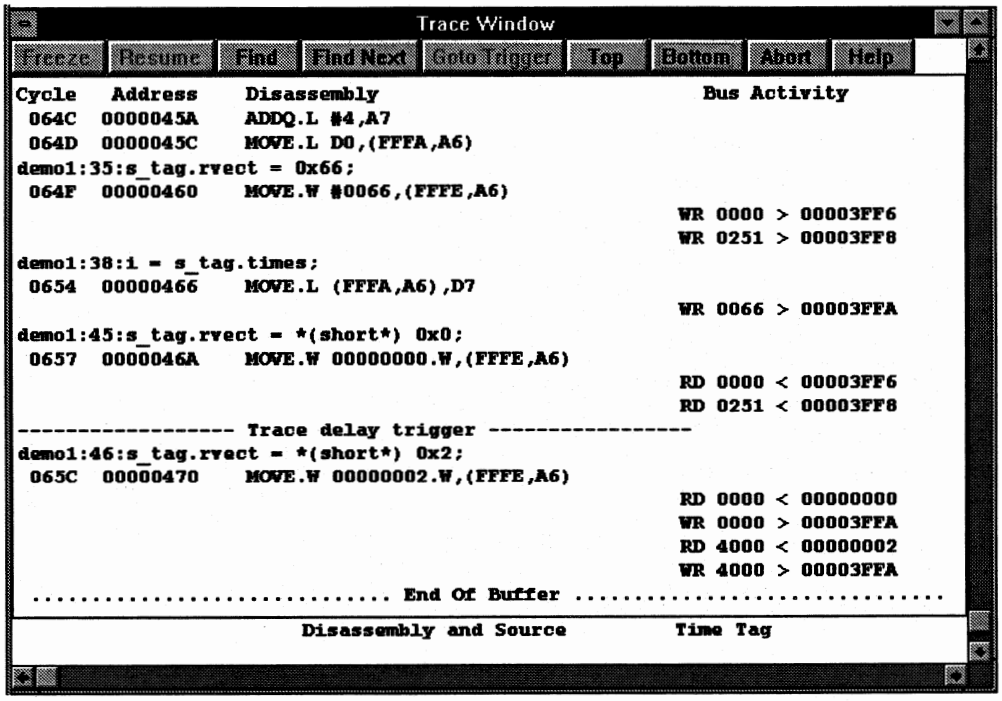
The status bits shown in the trace buffer are defined as follows when using a 68340 *Lite* unit:

DSSZ	DSACK1', DSACK0', SIZ1, SIZ0
FC	FC3, FC2, FC1, FC0
RSB	R/W', SHOW, BERR'
CS0-CS3	CS0'(AVEC'), CS1'(IRQ1'/PORTB1), CS2'(IRQ2'/PORTB2), CS3'(IRQ4'/PORTB4)

Note: () is alternate function depending on how pin is programmed

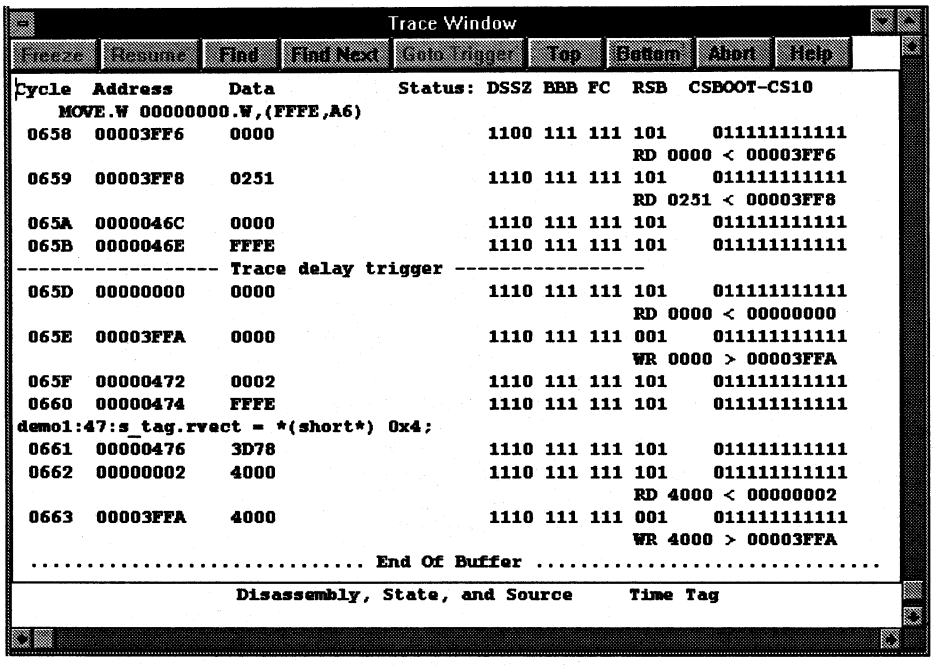
Disassembly/Source

The trace buffers will show both disassembly and source information which includes cycle, address, module name, line number of the module, the actual high-level line of code, and bus activity. All state specific information will be suppressed. To select this mode, choose 'Trace/View/Disassembly/Source'.



Disassembly/State/Source

The trace buffers will show all available information while in this mode which includes cycle, address, data, status bits, bus activity, module name, line number of the module, and the actual high-level line of code. No information is suppressed in this mode. To select this mode, choose 'Trace/View/Disassembly/State/Source'.



The status bits shown in the trace buffer are defined as follows when using a 68331, 68332, or 68HC16 *Lite* unit:

DSSZ	DSACK1'(PORTE1), DSACK0'(PORTE0), SIZ1(PORTE7), SIZ0(PORTE6)
BBB	BR'(CS0'), BG'(CS1'), BGACK'(CS2')
FC	FC2(CS5'), FC1(CS4'), FC0(CS3')
RSB	R/W', SHOW, BERR'
CSBOOT-CS0	CSBOOT', CS0'(BR'), CS1'(BG'), CS2'(BGACK'), CS3'(FC0), CS4'(FC1), CS5'(FC2), CS6'(A19), CS7'(A20), CS8'(A21), CS9'(A22), CS10'(A23)

The status bits shown in the trace buffer are defined as follows when using a 68340 *Lite* unit:

DSSZ	DSACK1', DSACK0', SIZ1, SIZ0
FC	FC3, FC2, FC1, FC0
RSB	R/W', SHOW, BERR'
CS0-CS3	CS0'(AVEC'), CS1'(IRQ1'/PORTB1), CS2'(IRQ2'/PORTB2), CS3'(IRQ4'/PORTB4)

Note: () is alternate function depending on how pin is programmed

Freeze Trace Operation

The 'Freeze' feature allows you to view the trace buffer without stopping emulation. While "freeze" is active, functions such as tracing, breaking, and triggering are disabled. The following sections give an overview of how to use the Freeze/Resume feature.

Freezing The Trace

The 'Freeze' command can be issued in one of three ways. From the Trace window, you can either choose 'Trace/Freeze' or press the 'Freeze' button. The 'Freeze' command can also be issued from the 'Run Status' window. After the 'Freeze' command is issued, you can then view the trace buffer. If the trace trigger and associated delay condition have occurred, the freeze trace will be the same information as contained in the trigger trace. If these conditions have not occurred, the freeze trace will show a 'snapshot' of the last 4K cycles of information executed before the 'Freeze' command was given.

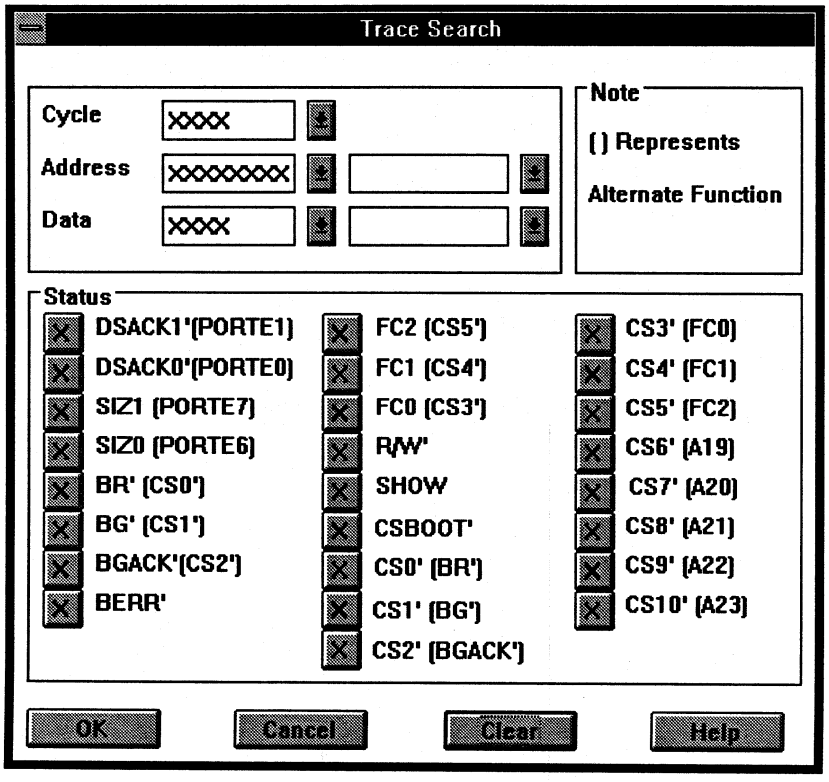
Resuming The Trace

The 'Resume' command can be issued in one of three ways. From the Trace window, you can either choose 'Trace/Resume' or press the 'Resume' button. The 'Resume' command can also be issued from the 'Run Status' window. After the 'Resume' command is given, the trace buffer will again begin recording trace information.

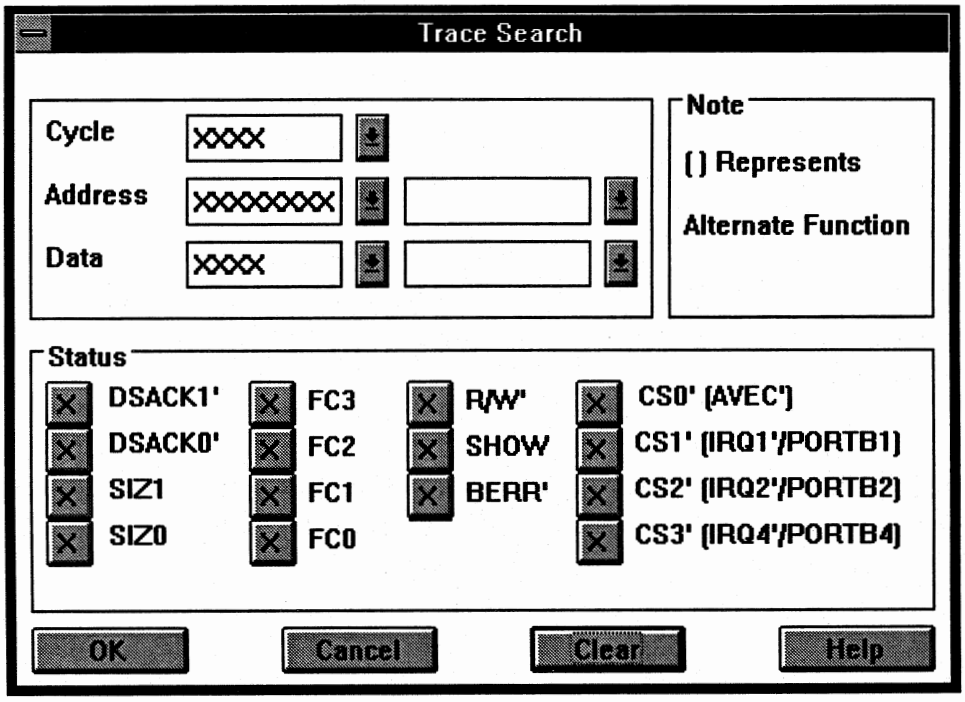
Searching The Trace

You can search the trace buffer for a particular condition. To search the trace buffer, choose "Trace/Find" or press the "Find" button. This will open the Trace Search window. You can then enter the pattern of information to be searched for. The search condition can be any combination of cycle, address, data, and status bits.

The 68331/68332/68HC16 *Lite* Trace Search window has the following format:



The 68340 *Lite* Trace Search window has the following format:



The Cycle, Address, and Data values can be changed by selecting the desired field and entering a value for that field. Previously entered values can be obtained by using the **Scroll Box** button to reveal and then select the desired value. Symbols can be entered for the Address and Data fields by pressing the right mouse button while the cursor is in a symbol entry field located to the right of both fields. The status bit values can be changed by placing the mouse pointer on the desired status bit and pressing the left mouse button. This will toggle through the choices for the status bits. The choices are 0, 1, and 'X' where 'X' is the don't care condition.

After the search function is started, the entire trace buffer will be searched and the first cycle where the search condition is met will be shown at the top of the trace buffer. To search for the next occurrence of the defined pattern, choose "Search/Repeat Last Search" or press the **Find Next** button.

68331/68332/68HC16 Lite Trace Search Status Bits

A description of each of the status bits for the 68331, 68332, and 68HC16 *Lite* is given below:

DSACK0' and DSACK1':
The DSACK0'(PORTE0) and DSACK1'(PORTE1) status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1). Note that () is the alternate function depending on how the pin is programmed.

SIZ0 and SIZ1:
The SIZ0(PORTE6) and SIZ1(PORTE7) status bits allow you to define a condition to be when each of these signals takes on a certain value. Note that () is the alternate function depending on how the pin is programmed.

BR',BG', and BGACK':
The BR'(CS0'), BG'(CS1'), and BGACK'(CS2') status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1). Note that () is the alternate function depending on how the pin is programmed.

FC0,FC1, and FC2:
The FC0(CS3'), FC1(CS4'), and FC2(CS5') status bits allow you to define a condition to be when the function codes take on a certain value. Note that () is the alternate function depending on how the pin is programmed.

R/W', SHOW, and BERR':
The R/W' and BERR' status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1). Note that () is the alternate function depending on how the pin is programmed. The SHOW status bit allows you to define a condition to be when a SHOW CYCLE occurs (SHOW = 1) or when a SHOW CYCLE does not occur (SHOW = 0).

CSBOOT', CS0' thru CS10':
The CSBOOT', CS0' thru CS10' status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1). Note that () is the alternate function depending on how the pin is programmed.

68340 Lite Trace Search Status Bits

A description of each of the status bits for the 68340 *Lite* is given below:

DSACK0' and DSACK1':
The DSACK0' and DSACK1' status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1).

SIZ0 and SIZ1:
The SIZ0 and SIZ1 status bits allow you to define a condition to be when each of these signals takes on a certain value.

FC0,FC1, FC2, and FC3:
The FC0, FC1, FC2, and FC3 status bits allow you to define a condition to be when the function codes take on a certain value.

R/W', SHOW, and BERR':
The R/W' and BERR' status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1). The SHOW status bit allows you to define a condition to be when a SHOW CYCLE occurs (SHOW = 1) or when a SHOW CYCLE does not occur (SHOW = 0).

CS0' thru CS3':
The CS0' (AVEC'), CS1' (IRQ1'/PORTB1), CS2' (IRQ2'/PORTB2), and CS3' (IRQ4'/PORTB4) status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1). Note that () is the alternate function depending on how the pin is programmed.

Trace Search Window Buttons

A description of the functions of the buttons available in this window are as follows:

0, 1, X

The **0, 1, X** button will toggle through the available choices for the status bits. The status bit values can be changed by placing the mouse pointer on the desired status bit and pressing the left mouse button. This will toggle through the choices for the status bits. The choices are **0**, **1**, and **X** where **X** is the don't care condition.

Ok

The **Ok** button is used to accept the changes made in the search pattern and to begin the search.

Cancel

The **Cancel** button will cancel the changes made in the search pattern and will close the Trace Search window.

Clear

The **Clear** button will set all entries in this window to "X"s (don't care).

Help

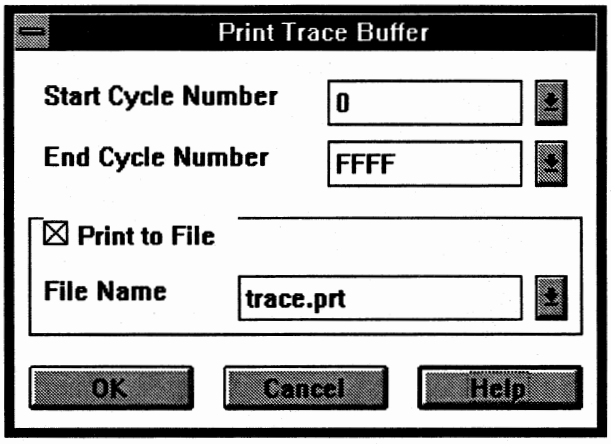
The **Help** button will invoke help for this window.

Scroll Box

The **Scroll Box** button will allow you to select a previously entered value for that field.

Printing The Trace

To print the trace buffer, choose "File/Print". This will open the 'Trace Print' window. The format of this window is shown as follows:



You can define the beginning and ending cycle number which determines how much of the trace information is output. If the 'Print to File' box is checked, you can print the trace information to a file instead of a printer. Simply enter the filename in the 'File Name' field and then press the OK button. If the 'Print to File' box is not checked, the trace output will be sent to the default printer. To define the printer to be used, choose "File/Printer Setup".

Print Trace Buffer Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will begin the trace printing operation.

Cancel

The **Cancel** button will abort the trace printing operation.

Help

The **Help** button will invoke help for this window.

Trace Window Buttons

A description of the functions of the buttons available in this window are as follows:

Freeze

The **Freeze** button allows you to view the trace while the emulator continues to run.

Resume

The **Resume** button should be pressed when the trace is in freeze mode in order to return to the normal mode of emulation.

Find

The **Find** button should be pressed when you wish to search for a particular pattern in the trace buffer.

Find Next

The **Find Next** button will find the next occurrence of the defined search pattern.

Goto Trigger

The **Goto Trigger** button will find the trace trigger if it exists in the trace buffer.

Top

The **Top** button will take the display to the top of the trace buffer.

Bottom

The **Bottom** button will take the display to the bottom of the trace buffer.

Abort

The **Abort** button will stop the trace display from scrolling after the current scroll operation is completed. This button is handy if you have entered multiple page up or page down operations and then decide to cancel the scrolling operation.

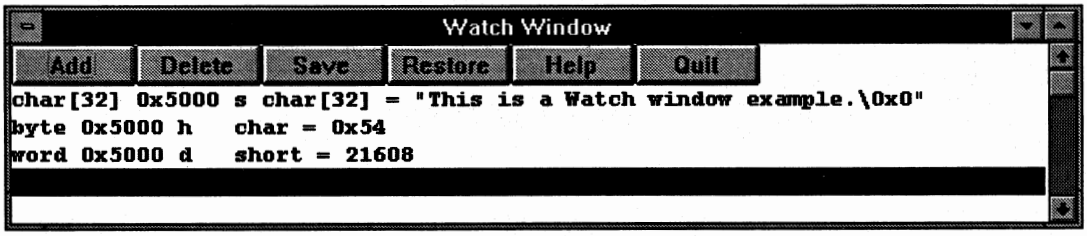
Help

The **Help** button will invoke help for this window.

Working With The Watch Window

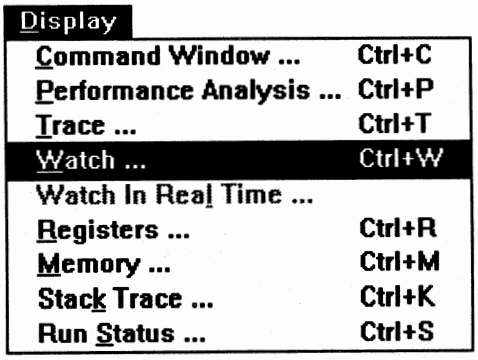
The Watch window allows you to "watch" certain memory locations or variables. You can choose the variable type, address, and display type that is shown in the Watch window. Data structure information along with the display of local variables is supported when Archimedes, Software Development Systems (SDS), or any other compiler that can produce IEEE-695 or Common Object File Format (COFF) output files is used. Watch windows can be saved to disk and loaded again at a later date. Most variable types are supported and can be displayed.

The Watch window is updated each time emulation is terminated through a breakpoint, stop command, or reset command. It will also be updated during single step operations. An example Watch window is shown below:



Opening The Watch Window

To open the Watch window, choose 'Display/Watch':



An alternate method of opening the Watch window is to use the accelerator key combination Ctrl+W.

Adding A Variable To The Watch Window

Variables can be added to the Watch window by clicking on the **Add** button or by entering in the variable information manually. Each of these methods will be reviewed in the following sections.

Entering Watch Variables Manually From The Watch Window

The Watch window allows the user to enter watch variables manually by clicking on a line in the watch window and typing in the desired type, address, and display type information for the variable to be "watched" in the format of:

type address display_type

where 'type' is the variable type, 'address' is the address (or symbol name) of the variable, and 'display_type' is the format you wish the variable displayed in. A space must be used between each of these entries. After entering in the desired information, press the 'Enter' key to store the entry into the Watch window. The current value of the newly entered watch variable will then be displayed.

Probably the easiest way to enter a watch variable that is a symbol name is to press the right mouse button while in the Watch window. This will invoke the Symbol Selection window from which a symbol can be chosen by clicking on the desired variable and then pressing the OK button. Once the symbol has been selected, it will automatically be added to the watch variable list.

Each of the manual entry fields (type, address, display_type) are described in the sections that follow.

Type

The type field is used to tell the Watch window what type of variable is being specified. Any type can include an '*' after the type to denote a pointer.

The type field is used to tell the Watch window what type of variable is being specified. Any type can include an '*' after the type to denote a pointer. Some examples follow:

char	Variable at the address is of type char
char*	Value is a pointer to a variable of type char
char**	Value is a pointer to a pointer that points to a variable of type char

The following types are available and should be entered as shown:

byte	double
char	unsigned byte
int (or integer)	unsigned char
short	unsigned int (or integer)
word	unsigned short
long	unsigned word
address	unsigned long
float	

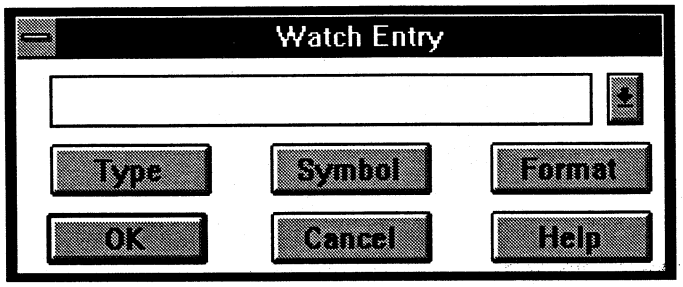
Address
The address field is used to tell the Watch window the address of the data to be watched. This field can be entered in decimal (default) or in hexadecimal by preceding the address with "0x" or "0X". On some emulators, you can specify an override designator as part of an address. Overrides may precede the address in the form {override}address. For example, if you wanted to watch an integer in the supervisor/data space of a 68000 based system at location 5000 hex and have it displayed as a decimal value, you should enter "integer {SD}0x5000 d".

Display_Type
The display type field is used by the Watch window to determine the way in which the data will be displayed. The default display type is decimal. All types must be preceeded with "%", the case is ignored. An example of the available display types follows:

%h (or x)(Hex)
%d (Decimal)
%o (Octal)
%b (Binary)
%c (Char)
%s (String)
%u (Unsigned Decimal)
%f (Float)

Using The Watch Entry Window To Enter Watch Variables

To add a variable to the Watch window using the 'Watch Entry' window, press the 'Add' button. This will open the 'Watch Entry' window:



While in this window, you can enter a watch variable by using the buttons shown in this window or by typing in the information manually. The following sections explain how to enter a variable using the buttons and manually.

Adding A Watch Variable Manually

The 'Watch Entry' window allows you to enter watch variables manually. From here, you can enter the type, address, and display type information for the variable to be "watched" in the format:

type address display_type

where 'type' is the variable type, 'address' is the address (or symbol name) of the variable, and 'display_type' is the format you wish the variable displayed in. A space must be used between each of these entries. If you realize you have made a mistake after you have chosen OK, you can edit any previous entry by pressing the **Scroll Box** button located next to the variable entry field. This allows you to load a previous entry into the 'Watch Entry' window.

If the variable to be watched is a pre-defined symbol, you can bypass the above steps by typing the name of the symbol in the watch entry field and pressing the OK button. SourceGate will search the symbol table information and display the variable in its proper form.

Probably the easiest way to enter a watch variable that is a symbol name is to press the right mouse button while in the Watch window. This will invoke the Symbol Selection window from which a symbol can be chosen by clicking on the desired variable and then pressing the OK button. Once the symbol has been selected, it will automatically be added to the watch variable list.

Each of the manual entry fields (type, address, display_type) are described in the sections that follow.

Type
The type field is used to tell the Watch window what type of variable is being specified. Any type can include an '*' after the type to denote a pointer.

Some examples follow:

char	Variable at the address is of type char
char*	Value is a pointer to a variable of type char
char**	Value is a pointer to a pointer that points to a variable of type char

The following types are available and should be entered as shown:

byte	double
char	unsigned byte
int (or integer)	unsigned char
short	unsigned int (or integer)
word	unsigned short
long	unsigned word
address	unsigned long
float	

Address
The address field is used to tell the Watch window the address of the data to be watched. This field can be entered in decimal (default) or in hexadecimal by preceding the address with "0x" or "0X". On some emulators, you can specify an override designator as part of an address. Overrides may precede the address in the form {override}address. For example, if you wanted to watch an integer in the supervisor/data space of a 68000 based system at location 5000 hex and have it displayed as a decimal value, you should enter "integer {SD}0x5000 d".

Display_Type
The display type field is used by the Watch window to determine the way in which the data will be displayed. The default display type is decimal. All types must be preceeded with "%", the case is ignored. An example of the available display types follows:

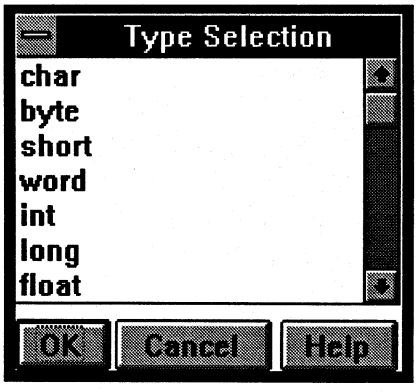
%h (or x)(Hex)
%d (Decimal)
%o (Octal)
%b (Binary)
%c (Char)
%s (String)
%u (Unsigned Decimal)
%f (Float)

Adding A Watch Variable Using Buttons

The Watch Entry window buttons can be used to enter the information for a watch variable definition. The **Type**, **Symbol**, and **Format** buttons invoke additional windows from which watch variable parameters can be defined. An explanation on how to use each of these buttons is defined in the sections that follow. **It should be noted that, if you are using buttons to enter watch variables, you should not enter any parameters manually until all of the desired buttons have been pressed. If you previously entered information manually and then press a button, the manually entered information will be deleted. To mix manual and button entries, always use the buttons first and then edit the watch entry line manually.**

Type

The **Type** button will open the 'Type Selection' window as shown below:



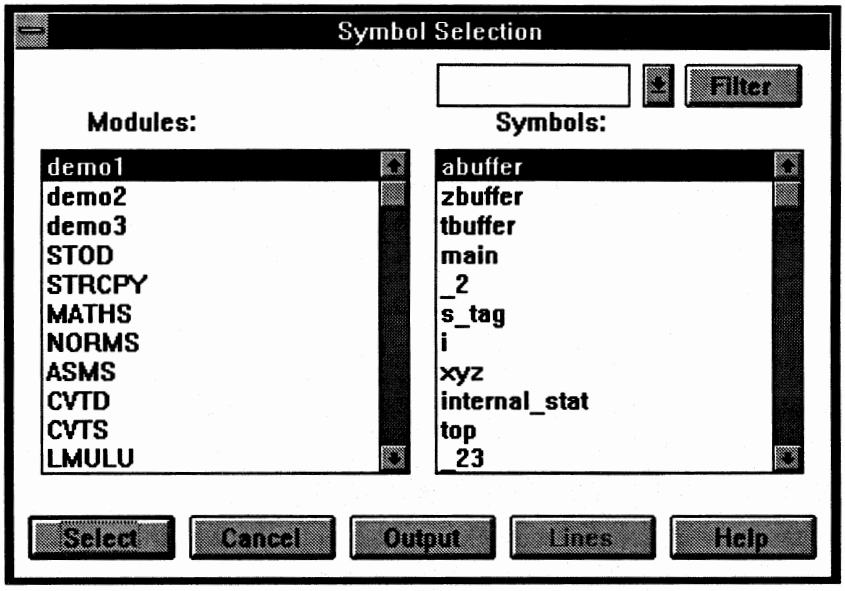
From this window, you can choose most of the available variable types. Additional types can be entered manually. To choose a type, click on it and then press the **Ok** button. The selected type will then be shown in the 'Watch Entry' window. The pre-defined choices are as follows:

char
byte
short
word
int
long
float
double
address
char *
int *
long *

If you decide not to enter a type, press the **Cancel** button. To obtain help for this window, press the **Help** button.

Symbol

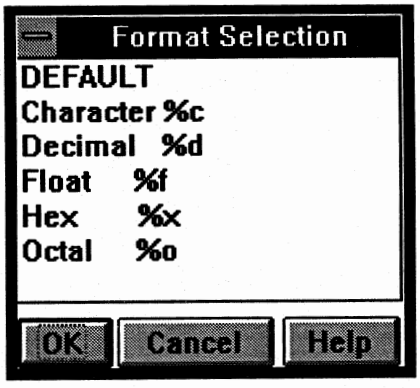
The **Symbol** button will open the 'Symbol Selection' window as shown below:



From this window, you can select one of the pre-defined symbols as the variable to be watched by highlighting it and pressing the **Select** button. The selected symbol will then appear in the 'Watch Entry' window. For additional information on using this window, see ***Working With Symbols*** in Chapter 2.

Format

The **Format** button will open the 'Format Selection' window as shown below:



The Format Selection window allows you to choose a pre-defined variable display format by using the mouse pointer and left mouse button to highlight the desired format. Pressing the **Ok** button will load the defined format into the 'Watch Entry' window. The output value for this watch variable will be in the specified format when displayed in the Watch window. Note that DEFAULT values are set to the most common format for the defined type (e.g. pointers will default to hex, characters to ASCII, integers to decimal, etc.). If you decide not to enter a display type, press the **Cancel** button. To obtain help for this window, press the **Help** button.

Ok
The **Ok** button will accept the defined watch entry and place the variable in the Watch window. If there are any formatting errors encountered with the entry, they will be displayed in the Watch window. To edit a previously entered watch entry, press the **Scroll Box** button located next to the watch entry field and choose one of the entries shown. You can then edit it.

Cancel
The **Cancel** button will cancel the watch entry operation and will close the 'Watch Entry' window.

Help
The **Help** button will invoke help for this window.

Scroll Box
The **Scroll Box** button will allow you to choose a previously entered watch entry.

Deleting A Variable From The Watch Window
To delete a previously entered watch variable from the Watch window, highlight the variable by clicking on it, then press the **Delete** button. The variable will then be removed from the Watch window.

Saving A Watch Window
To save a Watch window's contents for future reference, press the **Save** button. Then enter the name of the file to save the Watch window to. Once this is done, the variables defined in the current Watch window will be saved to the named file. To re-load this Watch window at a later date, press the **Restore** button and choose the name of the file that you saved. The variables saved in this file will then be displayed with their current values.

Restoring A Saved Watch Window
Once you use the **Save** button to save a Watch window to a file, you can restore this saved Watch window by pressing the **Restore** button and then choosing the filename of the saved file. The variables saved in this file will then be displayed with their current values.

Watch Window Buttons
A description of the functions of the buttons available in this window are as follows:

Add
The **Add** button will add a watch variable to the Watch window.

Delete
The **Delete** button will delete a watch variable from the Watch window. If the highlight line is on the blank line immediately after the last watch entry, pressing this button will delete the last watch entry. This is handy if you wish to delete multiple watch entries. You can also use the Ctrl-Shift buttons, along with the mouse, to highlight multiple watch entries and then delete them all by pressing the **Delete** button.

Save
The **Save** button will save a Watch window's contents to a file.

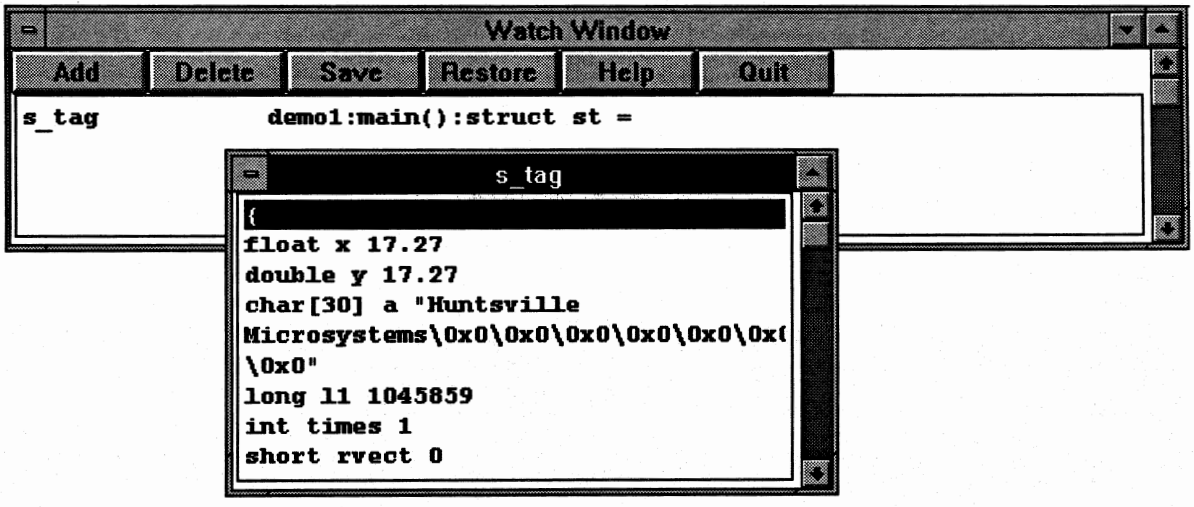
Restore
The **Restore** button will restore a previously saved Watch window's contents to the Watch window.

Close
The **Close** button will close the Watch window.

Help
The **Help** button will invoke help for this window.

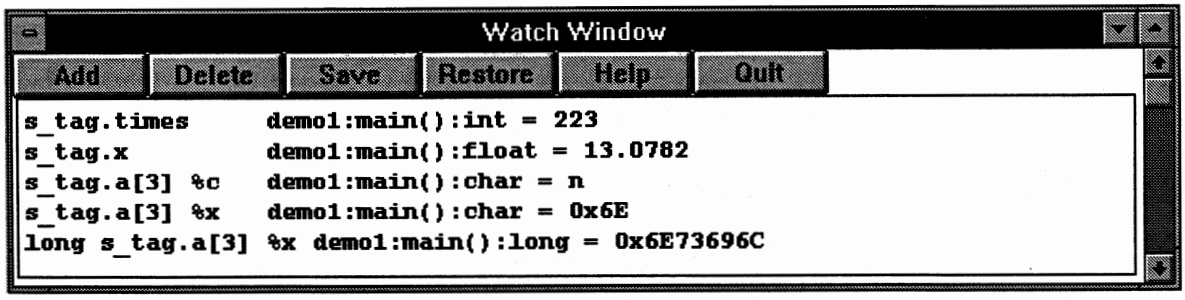
Watch Window Examples
The following examples were included in order to help give you a better understanding of how to use the Watch window.

Watching A Structure
To watch an entire structure, enter the name of the structure as a watch variable. A separate window showing the contents of each element of the structure will be shown if the structure's information is too long to be shown in the Watch window:



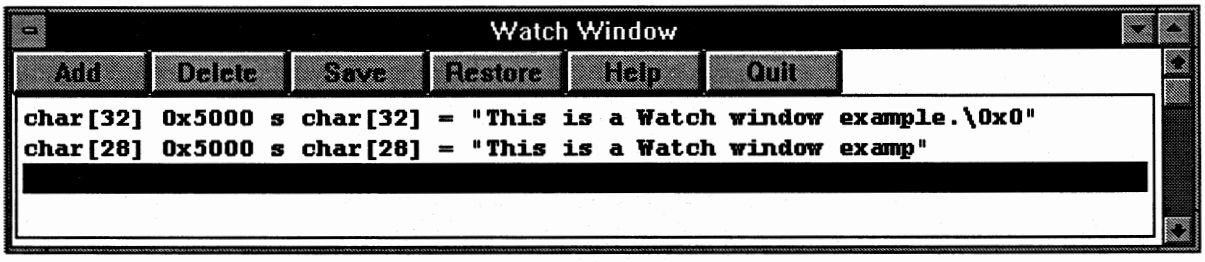
Watching An Element Of A Structure

If you wish to watch certain elements of a structure, this can be accomplished by entering "structure.element":



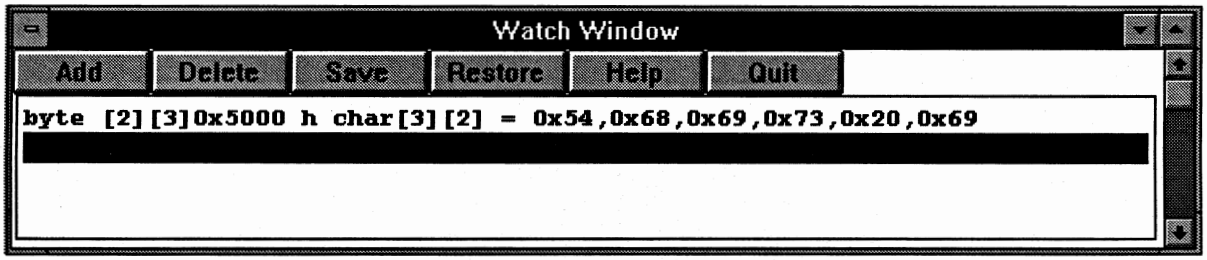
Watching A One-Dimensional Array

To watch a one-dimensional array, enter the array size. If you wish to watch certain elements of an array, this can be accomplished by entering "array_name[element]". The following is an example of a character array:



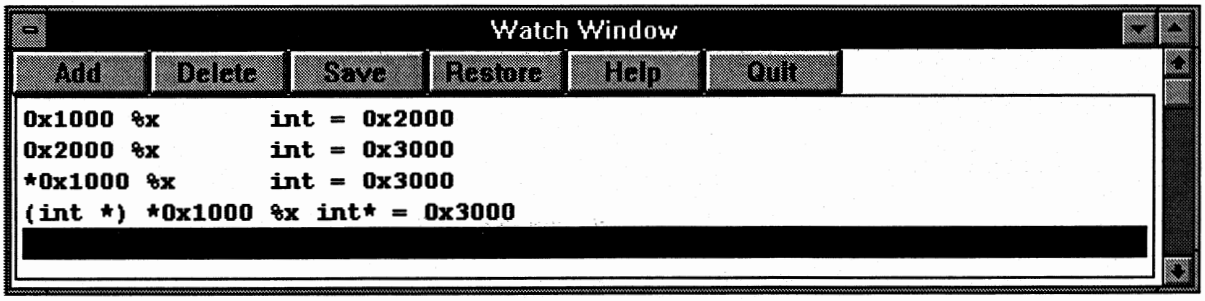
Watching A Multi-Dimensional Array

For multi-dimensional arrays, type "array_name[a][b]" to watch particular elements. Arrays will be displayed in a separate "Watch Output" window if the output if too large to be shown in the Watch window.



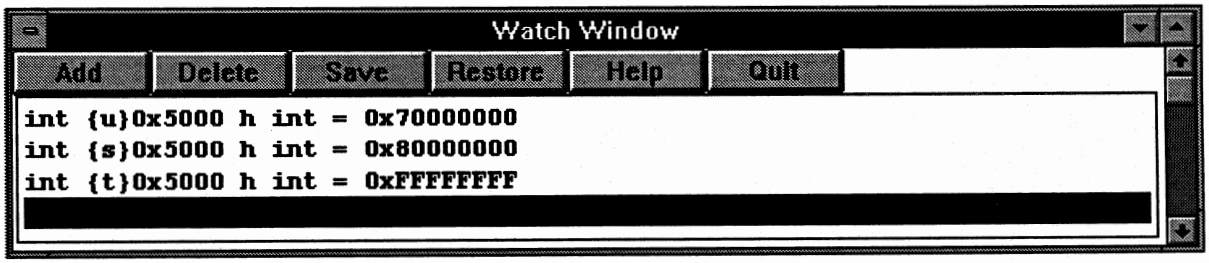
Watching Pointers

Pointers can be watched by preceeding an address with the pointer designator "*".



Using Overrides In The Watch Window

Overrides can be used to watch different memory spaces. In the example shown below, address 0x5000 in a 68030 system is being watched. The 68030 has separate Supervisor and User memory spaces. The {s} override is a 68030 override which specifies Supervisor memory space and the {u} override is a 68030 override which specifies User memory space. The {t} override specifies target memory. Notice that, using overrides, the contents of address 0x5000 is shown to be different for each memory space, which is to be expected since there are actually three distinct 0x5000 addresses. Without the use of overrides, only one space could be shown at a time.

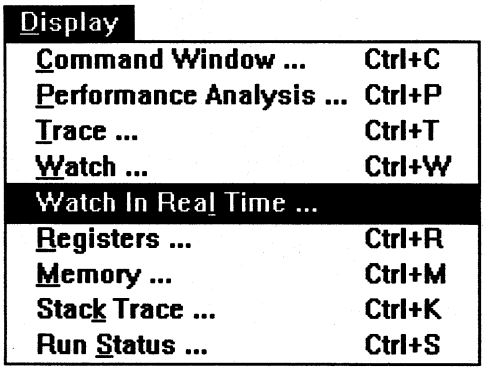


Working With The Watch In Real Time Window

The Watch In Real Time window allows you to watch shadow RAM memory locations update in real-time while emulation is in progress. You should reference the proceeding section, ***Working With The Watch Window*** for information on using this window as it's operation is similar to that of the Watch window.

Opening The Watch In Real Time Window

To open the Watch In Real Time window, choose 'Display/Watch In Real Time':

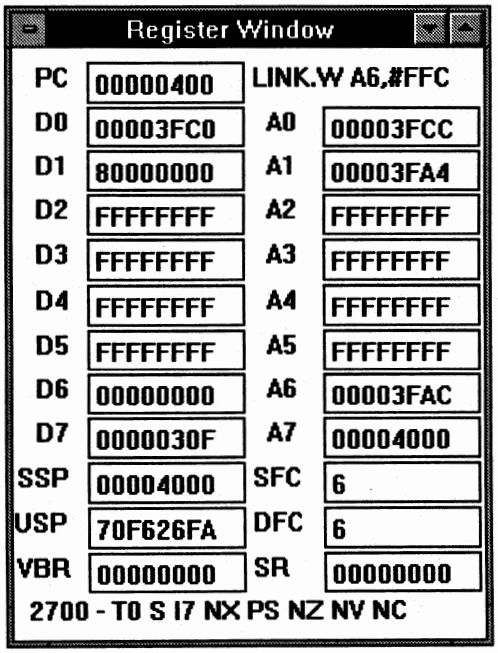


Again, reference the preceding section *Working With The Watch Window* for additional information on using this window.

Working With The Register Window

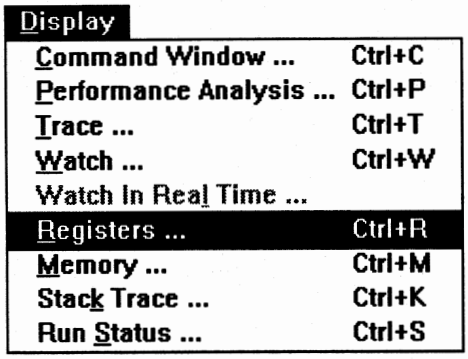
The Register window shows the contents of the processor's registers including the status of the flags and the current disassembly mnemonic being pointed to by the program counter register. The register values are always accurate whenever emulation is not in progress, and are the last known value whenever emulation is in progress. Registers can be altered in the register display window by selecting the desired register data field and then and entering a new value.

The Register window is automatically updated whenever the emulator is single-stepped, a breakpoint occurs, a stop command is executed, or the emulator is reset. The registers shown in the Register window are dependent on the emulator being used. An example *Lite* Register window is shown below:


Register Window
PC 00000400 LINK.W A6,#FFC
D0 00003FC0 A0 00003FCC
D1 80000000 A1 00003FA4
D2 FFFFFFFF A2 FFFFFFFF
D3 FFFFFFFF A3 FFFFFFFF
D4 FFFFFFFF A4 FFFFFFFF
D5 FFFFFFFF A5 FFFFFFFF
D6 00000000 A6 00003FAC
D7 0000030F A7 00004000
SSP 00004000 SFC 6
USP 70F626FA DFC 6
VBR 00000000 SR 00000000
2700 - T0 S I7 NX PS NZ NV NC

Opening The Register Window

To open the Register window, choose 'Display/Registers':



An alternate method of opening the Register window is to use the accelerator key combination Ctrl+R.

Modifying Register Values

To modify a register, move the insertion bar to the register to be changed and then press the left mouse button. You can then enter the desired information beginning at the insertion point. To delete the old contents of a register and add a completely new value, click and drag to highlight the entire contents of the register and then enter the new value.

Register Window Contents

The registers shown in the Register window are as follows:

Program Counter Register

The Program Counter is a 32-bit register that contains the address of the next instruction to be executed by the processor.

Program Counter Description Field

The Program Counter Description field is the field right next to the PC field. This field shows the assembled instruction pointed to by the PC and is automatically updated by the emulator. This field cannot be edited.

Status Register

The Status Register is a 16-bit register in which each bit is used to indicate the results of the instruction just executed. The user byte containing the condition codes is the only portion of the Status Register information available at the user privilege level, and it is referenced as the CCR in user programs. In the supervisor privilege level, software can access the full Status Register, including the interrupt priority mask (three bits) as well as additional control bits.

The individual condition codes are also displayed below the hexadecimal SR value. An explanation of each condition code is listed below:

SR Bit	Flag or Condition Code Name	Not Active Code	Active Code
15	Trace Enable T1	T0 or T1	T2 or T3
14	Trace Enable T0	T0 or T2	T1 or T3
13	Super/User State	U	S
12	Not Defined	--	--
11	Not Defined	--	--
10	Int Priority Mask 2		
9	Int Priority Mask 1	I0 through I7	
8	Int Priority Mask 0		
7	Not Defined	--	--
6	Not Defined	--	--
5	Not Defined	--	--
4	Extend	NX	XT
3	Negative	PS	NG
2	Zero	NZ	ZR
1	Overflow	NV	OV
0	Carry	NC	CY

Supervisor Stack Pointer Register

The Supervisor Stack Pointer is a 32-bit register that contains the address of the next free location on the stack if the SSP is the active stack pointer. The SSP is only accessible at the Supervisor privilege level. The SSP value can be modified by placing the cursor on the desired field and entering a new value.

User Stack Pointer Register

The User Stack Pointer is a 32-bit register that contains the address of the next free location on the stack if the USP is the active stack pointer.

Vector Base Register

The Vector Base Register is a 32-bit register that contains the base address of the exception vector table in memory. The displacement of an exception vector is added to the value in this register to access the vector table.

Data Registers D0-D7

Registers D0 - D7 are 32-bit data registers that are used for bit, bit-field, byte, word, long word, and quad word operations. These register values can be modified by placing the cursor on the desired field and entering a new value.

Address Registers A0-A7

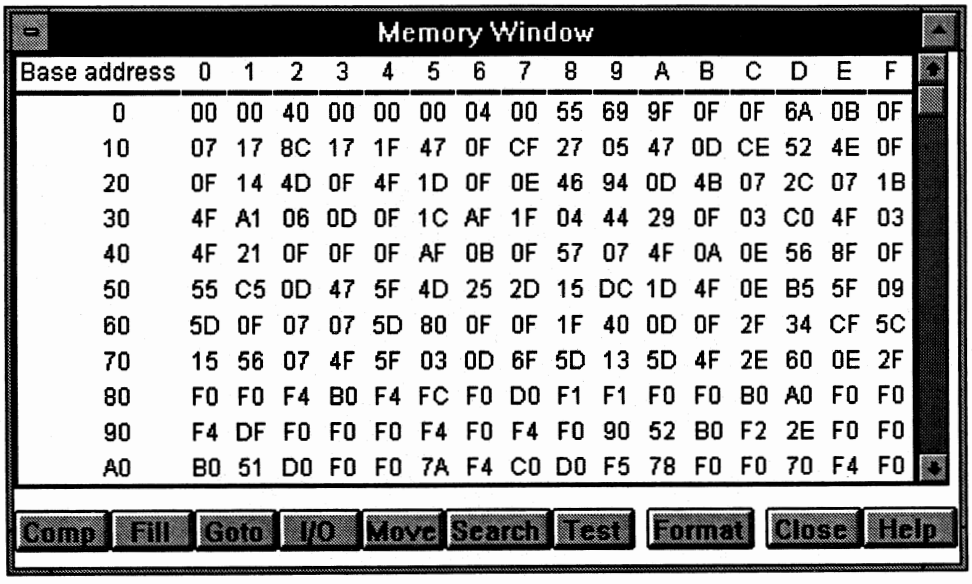
Registers A0 - A6 are 32-bit registers that may be used as software stack pointers or base address registers. Register A7 is a register designation that applies to the USP in the user privilege level, and to either the ISP or MSP in the supervisor privilege level.

Alternate Function Code Registers

The Alternate Function Code registers DFC and SFC contain 3-bit function codes. Function codes can be considered extentions of the 32-bit linear address that optionally provide as many as eight 4-gigabyte address spaces. Function codes are automatically generated by the processor to select address spaces for data and program at the user and supervisor privilege levels, and a CPU address space used for processor functions (for example, coprocessor communications). Registers DFC and SFC are used by certain instructions to explicitly specify the function codes for operations.

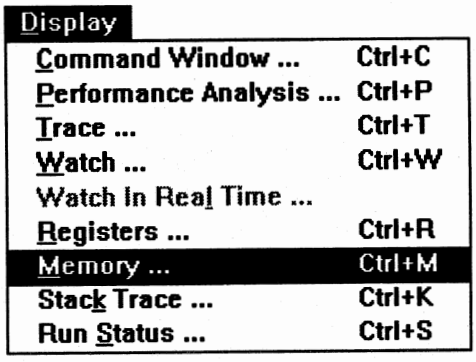
Working With The Memory Window

The memory window allows you to display and change individual memory locations. Additional features of this window allow you to compare blocks of memory, fill a memory block with a constant value, move a block of memory, search a block of memory for a particular value, and test a memory block among others. The Memory window has the following format:



Opening The Memory Window

To open the Memory window, choose 'Display/Memory':



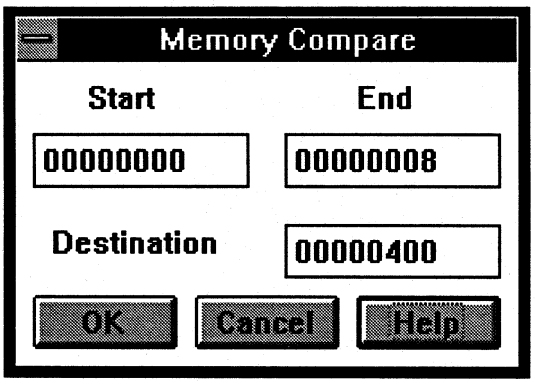
An alternate method of opening the Memory window is to use the accelerator key combination Ctrl+M.

Modifying Memory Values

To change a memory value, place the cursor on the value to be changed and press the left mouse button. To select the entire number, press the left mouse button and then drag until the whole number is highlighted. A new value can then be entered. After the desired value has been entered, click on another location or press one of the arrow keys to go to an adjacent memory location. This will load the value you just changed into memory at the specified address. Any values that you have changed will be shown in a different color for easy identification.

Comparing Memory Blocks

The Memory Compare window is invoked when the **Comp** button is pressed from within the Memory window. This window allows you to compare two memory blocks to see any differences between them. The Memory Compare window has the following format:



The 'Start' field defines the starting address of the first block, the 'End' field defines the ending address of the first block, and the 'Destination' field defines the starting address of the second block. When the **Ok** button is pressed, the first memory location in the first block is compared to the first location in the second block, etc., until the defined areas have been compared.

In the example above, the first block starts at address 0 and ends at address 8H. The second block begins at address 400H. When the **Ok** button is pressed, SourceGate II will begin comparing each location in the first block to the corresponding location in the second block.

Memory Compare Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will begin the memory compare operation.

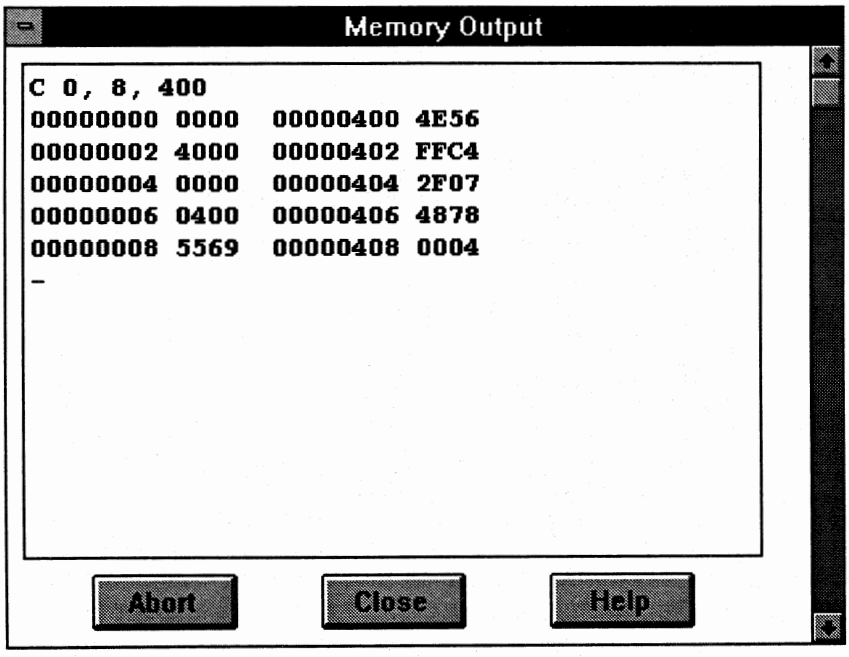
Cancel

The **Cancel** button will cancel the compare operation and will close the Memory Compare window.

Help

The **Help** button will invoke help for this window.

The results of the memory compare operation are shown in the Memory Output window. The Memory Output window has the following format:



The output of this window is of the format:

Addr1 Value1 Addr2 Value2

where:

Addr1 - is the address in block 1
Value1 - is the contents of Addr1
Addr2 - is the address in block 2
Value2 - is the contents of Addr2

If information in this format is shown, the comparison failed at these locations. If no information is shown, then all locations compared contain the same information.

Memory Output Window Buttons

A description of the functions of the buttons available in this window are as follows:

Abort

The **Abort** button will stop the output operation being performed. This could be useful if you compared a large memory block and wish to stop scrolling through the output.

Close

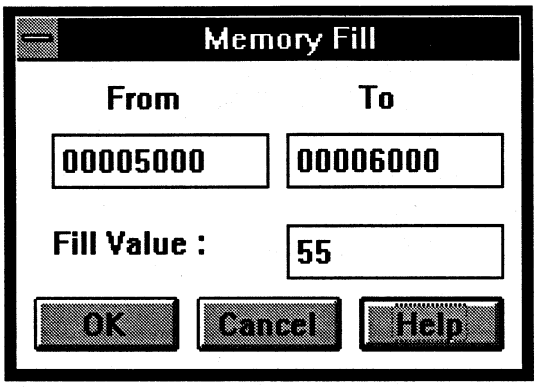
The **Close** button closes the Memory Output window.

Help

The **Help** button invokes help for this window.

Filling Memory With A Constant

The Memory Fill window is invoked when the **Fill** button is pressed from within the Memory window. This window allows you to fill a memory block with a constant value. The Memory Fill window has the following format:



The 'From' field defines the beginning address in memory where the fill will begin, the 'To' field defines the ending address in memory where the fill will stop, and the 'Fill Value' field defines the constant value that this memory block will be filled with. When the **Ok** button is pressed, the fill process will begin.

In the example above, memory from 5000H to 6000H will be filled with the value 55H when the **Ok** button is pressed.

Memory Fill Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will begin the fill operation.

Cancel

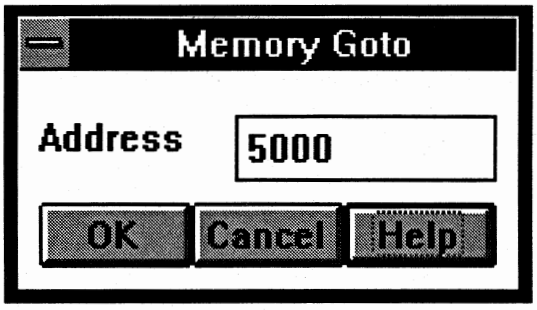
The **Cancel** button will cancel the fill operation and will close the Memory Fill window.

Help

The **Help** button will invoke help for this window.

Viewing A Specific Address In Memory

The Memory Goto window is invoked when the **Goto** is pressed from within the Memory window. This window allows you to define the base address of the memory locations shown in the Memory window. The Memory Goto window has the following format:



The 'Address' field defines the base address of the memory locations that will be shown in the Memory window. When the **Ok** button is pressed, the Memory window will be re-configured to show the memory locations beginning with the address defined here.

In the example above, the Memory window will show memory addresses starting at 5000H when the **Ok** button is pressed.

Memory Goto Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will make the Memory Window begin with the defined address.

Cancel

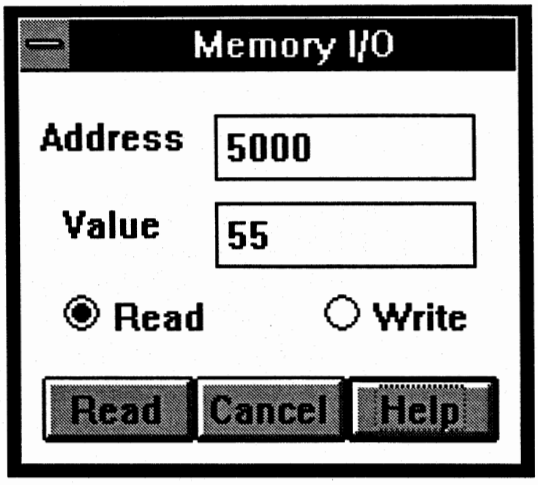
The **Cancel** button will cancel the goto operation and will close the Memory Goto window.

Help

The **Help** button will invoke help for this window.

Reading/Writing A Specific Memory Location

The Memory I/O window is invoked when the **I/O** button is pressed from within the Memory window. This window allows you to read or write a specific memory location. The Memory I/O window has the following format:



The 'Address' field defines the address to be read or written, the 'Value' field defines the value to be read or written, and the 'Read' and 'Write' check boxes define which operation is to be performed on the defined address.

In the example above, the value 55H has been read from address 5000H. This is the result of pressing the **Read** button. If a write operation was being performed, the **Read** button would change to **Write** and pressing the **Write** button would write 55H to the address 5000H.

Note that some emulators may not support this feature. If the I/O button is grayed in your Memory window, this feature is not available to you.

Memory I/O Window Buttons

A description of the functions of the buttons available in this window are as follows:

Read/Write

The **Read** button will read the contents of the defined address and shown the result in the 'Value' field. If a **Write** operation is selected, the **Read** button will change to **Write** and pressing this button will write the value defined in the 'Value' field to the defined address.

Cancel

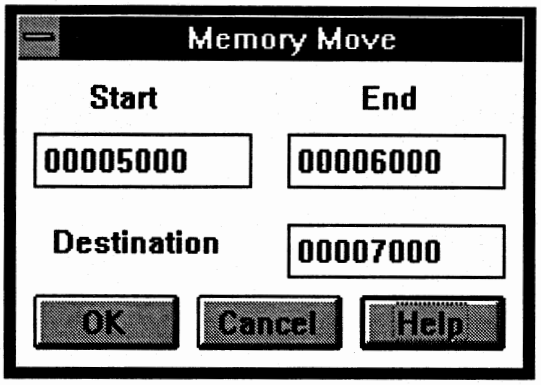
The **Cancel** button will cancel the I/O operation and will close the Memory I/O window.

Help

The **Help** button will invoke help for this window.

Moving Memory Blocks

The Memory Move window is invoked when the **Move** button is pressed from within the Memory window. This window allows you to move a block of memory to another location in memory. The Memory Move window has the following format:



The 'Start' field defines the starting address of the block to be moved, the 'End' field defines the ending address of the block to be moved, and the 'Destination' field defines the location in memory where the block is to be moved to. When the **Ok** button is pressed, the contents of the first memory location in the block to be moved is moved to the location specified in the 'Destination' field, the contents of the next location are moved to the location specified in the 'Destination' field plus one, etc., until the entire block has been moved.

In the example above, the block to be moved starts at address 5000H and ends at address 6000H. The destination for the block is 7000H. When the **Ok** button is pressed, SourceGate II will begin moving each location in the block to be moved to the corresponding location in the destination block.

Memory Move Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will begin the memory move operation.

Cancel

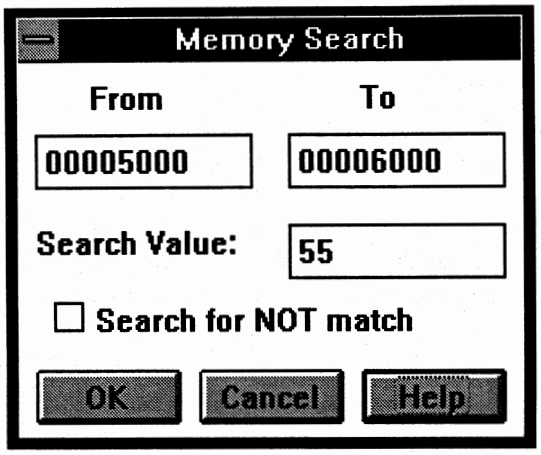
The **Cancel** button will cancel the move operation and will close the Memory Move window.

Help

The **Help** button will invoke help for this window.

Searching Memory Blocks

The Memory Search window is invoked when the **Search** button is pressed from within the Memory window. This window allows you to search a block of memory for a particular value or you can search a block of memory and display all locations that do not contain a specific value. This feature is useful for checking memory that is being initialized to a specific value. The Memory Search window has the following format:



The 'From' field is the beginning address where the search will begin, the 'To' field is the ending address where the search will end, and the 'Search Value' is the value to be searched for. The 'Search for NOT match' check box should be used to search for any occurrences in the defined block that do NOT contain the value defined in the 'Search Value' field.

In the example above, the block to be searched begins at location 5000H and ends at location 6000H. The value to be searched for is 55H. When the **Ok** button is pressed, SourceGate II will begin searching the specified block for the defined value.

Memory Search Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will begin the memory search operation.

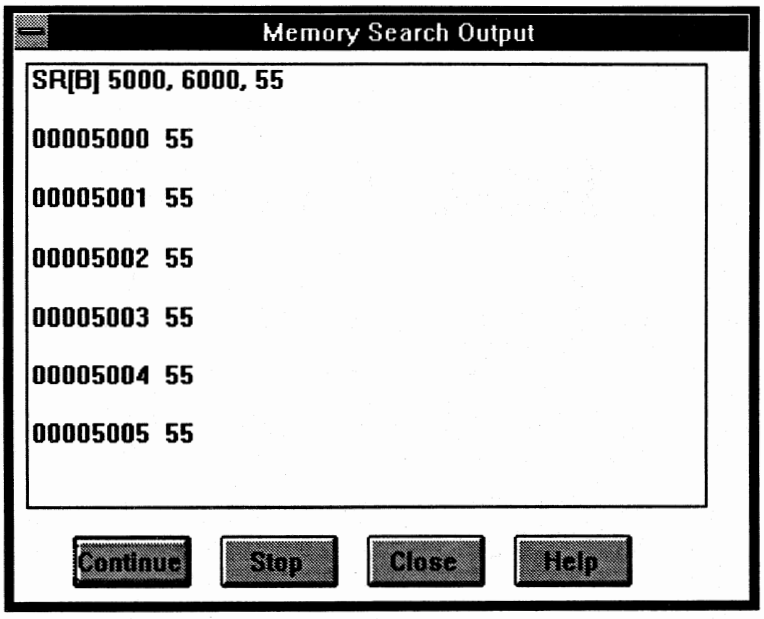
Cancel

The **Cancel** button will cancel the search operation and will close the Memory Search window.

Help

The **Help** button will invoke help for this window.

The results of the memory search operation are shown in the Memory Search Output window. The Memory Search Output window has the following format:



The output of this window is of the format:

Addr Value

where:

Addr - is the address of the search match
Value - is the value located at the address of the search match

If no information is shown, the search found no matches.

Memory Search Output Window Buttons

A description of the functions of the buttons available in this window are as follows:

Continue

Once a search match is found, searching will stop and the Memory Search Output window will shown the found location. The **Continue** button will allow you to begin searching for the next match.

Stop

The **Stop** button will stop the search operation.

Close

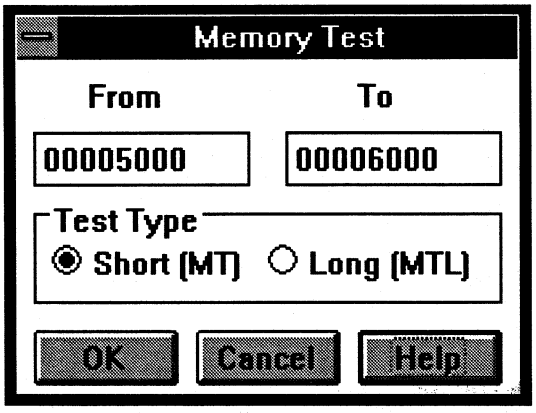
The **Close** button will close the Memory Search Output window.

Help

The **Help** button will invoke help for this window.

Testing Memory Blocks

The Memory Test window is invoked when the **Test** button is pressed from within the Memory window. This window allows you to test a block of RAM memory to see if it is functioning properly. The Memory Test window has the following format:



The 'From' field is the beginning address where the memory test will begin, the 'To' field is the ending address where the memory test will end, and the 'Test Type' check boxes determine the type of memory test performed. The 'Short [MT]' test will write 55H and AAH to each memory location in the defined block. The 'Long [MTL]' test will write a "walking one" bit pattern to each location in the defined block. The [MT] and [MTL] designators are the corresponding Command window commands that can be used to test memory. The 'Long' test is much more thorough than the 'Short' test though it takes longer to perform. Both tests are nondestructive, meaning that the original contents of the memory locations are preserved.

In the example above, the RAM block between 5000H and 6000H will be tested using the 'Short' memory test. When the **Ok** button is pressed, SourceGate II will begin testing the specified block of memory. **Note that only RAM locations can be tested due to the write requirements of the test.**

Memory Test Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will begin the memory test operation.

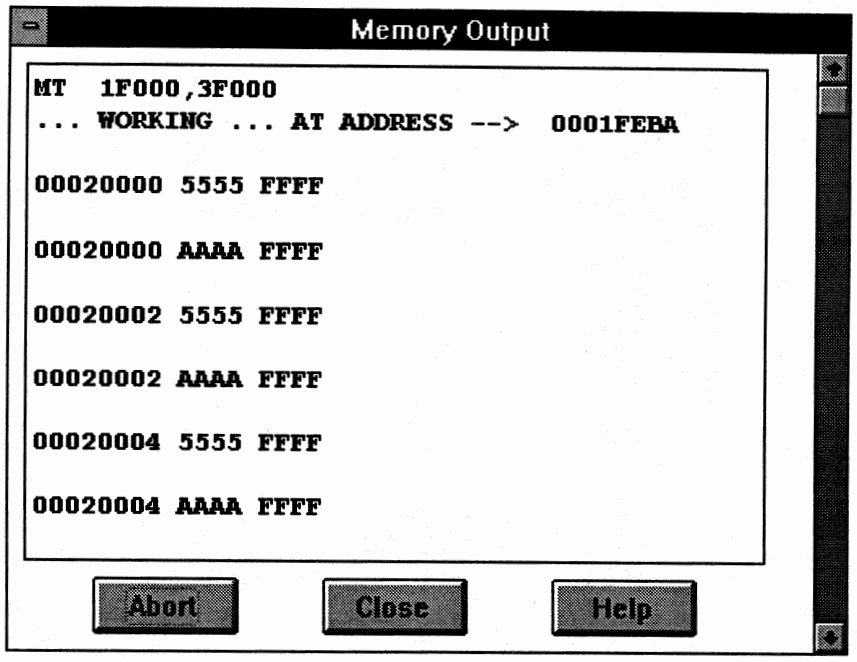
Cancel

The **Cancel** button will cancel the test operation and will close the Memory Test window.

Help

The **Help** button will invoke help for this window.

The results of the memory test operation are shown in the Memory Output window. The Memory Output window has the following format:



The output of this window is of the format:

Addr Value1 Value2

where:

Addr - is the address where memory failed
Value1 - is the value that was written to the failed block
Value2 - is the value that was read back from the failed block

If information in this format is shown, the comparison failed at these locations. If a message of the format "...WORKING... AT ADDRESS ==> XXXXXXXX" is shown, this shows that memory is being tested at the location shown and that no errors have yet been found.

Memory Output Window Buttons

A description of the functions of the buttons available in this window are as follows:

Abort

The **Abort** button will stop the output operation being performed. This could be useful if you tested a large memory block and wish to stop scrolling through the output.

Close

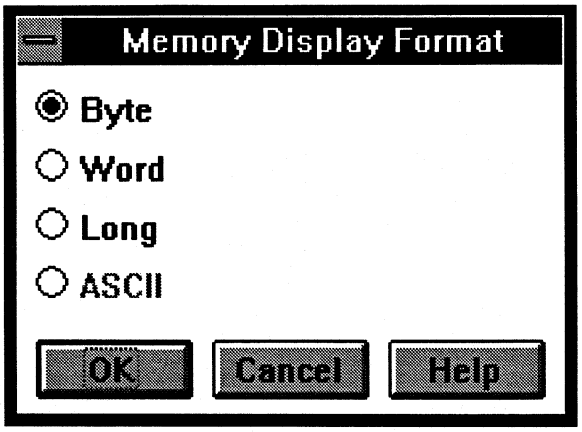
The **Close** button closes the Memory Output window.

Help

The **Help** button invokes help for this window.

Specifying The Display Format

The Memory Display Format window is invoked when the **Format** button is pressed from within the Memory window. This window allows you to change the display format of the Memory window. The Memory Display Format window has the following format:



To select a display format, you should place the mouse pointer on the desired format and then press the left mouse button. Once this is done and the **Ok** button is pressed, the Memory window will show memory values in the selected display format.

Memory Display Format Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will change the Memory window display to the selected format.

Cancel

The **Cancel** button will cancel the format change operation and will close the Memory Display Format window.

Help

The **Help** button will invoke help for this window.

Memory Window Buttons

A description of the functions of the buttons available in the Memory window are as follows:

Comp

The **Comp** button will allow you to compare two memory blocks and display any differences between them.

Fill

The **Fill** button will allow you to fill memory with a constant value.

Goto

The **Goto** button will allow you to define the starting address of the memory displayed in the Memory window.

I/O

The **I/O** button will allow you to read or write a specific memory location.

Move

The **Move** button will allow you to move a block of memory to another location.

Search

The **Search** button will allow you to search memory for a particular value.

Test

The **Test** button will allow you to test RAM locations to insure that they are functioning properly.

Format

The **Format** button will allow you define the format of the values shown in the Memory Window display. The available formats are 'Byte', 'Word', and 'Long'.

Close

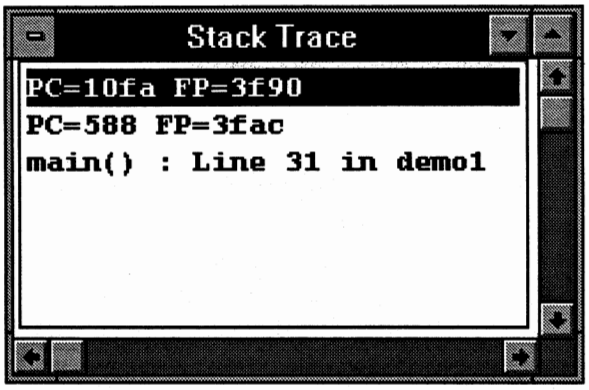
The **Close** button will close the Memory window.

Help

The **Help** button will invoke help for this window.

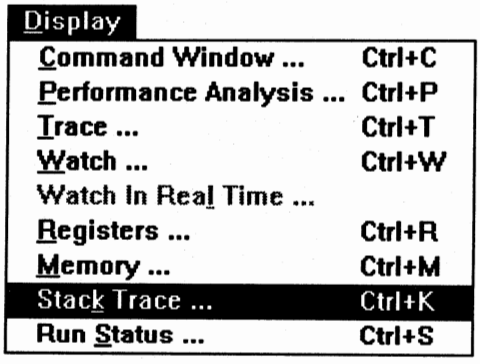
Working With The Stack Trace Window

The Stack Trace window shows the functions and local variables that are pushed onto the stack. The local variables passed and their values are displayed in their default base (hexadecimal, decimal, etc.). When stepping in assembly language, the Stack Trace window shows the current program counter and frame pointer values. The Stack Trace window has the following format:



Opening The Stack Trace Window

To open the Stack Trace window, choose 'Display/Stack Trace':



An alternate method of opening the Stack Trace window is to use the accelerator key combination Ctrl+K.

Stack Trace Window Contents

The Stack Trace window is automatically updated whenever the emulator is single-stepped, a breakpoint occurs, a stop command is executed, the emulator is reset, or memory is modified. The information shown in this window is the true information that is currently located on the stack. This provides you a way to track what is currently on the stack.

The information shown may be a combination of source line addresses and assembly addresses. The source line information will be in the format:

function (type var=value) : Line x in module

where:

function :	is the calling function
type:	is the variable type being passed
var:	is the variable name being passed
value:	is the value of the variable being passed
x:	is the line number of the calling module
module:	is the calling module

The assembly information will be in the format:

PC=X FP=Y

where:

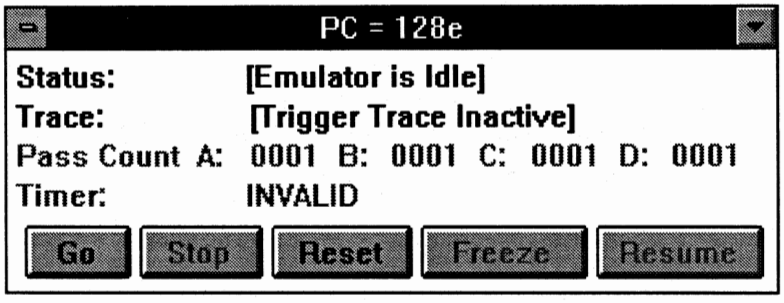
PC:	is the program counter
X :	is the value of the program counter
FP:	is the frame pointer
Y:	is the value of the frame pointer

An additional feature available from this window is the ability to call up a source or assembly window by double clicking on a line shown in the Stack Trace window or by pressing ENTER while the desired Stack Trace entry is highlighted. Once this is done, the source or assembly module associated with that Stack Trace entry is shown in a source or assembly window.

This version of SourceGate will display local variable values in this window for Archimedes and Software Development Systems (SDS) compilers and any compiler that can produce Common Object File Format (COFF) or IEEE-695 output files. Future releases may provide this feature for other compilers.

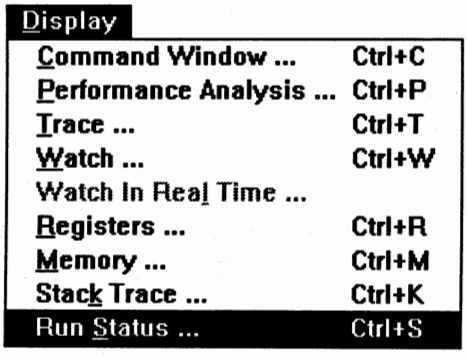
Working With The Run Status Window

The Run Status window allows you to view the emulation status using one window. The information shown in the border of this window includes the current program counter information or, if emulation is in progress, "RUNNING" is displayed. Inside of this window, the current emulation status (running or idle), and trace trigger status (active or inactive) are all displayed. In addition, there are buttons available that allow you to begin or stop emulation, reset the processor, and freeze or resume the trace buffers. The Run Status window has the following format:



Opening The Run Status Window

To open the Run Status window, choose 'Display/Run Status':



An alternate method of opening the Run Status window is to use the accelerator key combination Ctrl+S.

Run Status Window Fields

The Run Status window has several fields that show you the status information. These fields are information only fields and you cannot change them. An overview of these fields is given in the sections that follow:

Run Status 'Status' Field

The 'Status' field shows the status of the emulator. This field will either show "Emulator Running", "Emulator is Idle", or "Breakpoint Encountered" depending on the current conditions.

Run Status Trace Field

The 'Trace' field shows the status of the trace trigger. If the trace trigger is defined to be inactive, or if the conditions for the trace trigger have not yet been met, this field will show "Trace Trigger Inactive". If the trace trigger is defined to be active and the trace trigger conditions plus the defined delay value have been met, this field will show "Trace Trigger".

Run Status Window Buttons

A description of the functions of the buttons available in this window are as follows:

Go

The **Go** button begins emulation from the current program counter location shown in the border of the Run Status window.

Stop

The **Stop** button stops emulation.

Reset

The **Reset** button resets the emulator to its defined reset location.

Freeze

The **Freeze** button freezes the trace buffer. For additional information on using this button, see ***Freeze Trace Operation.***

Resume

The **Resume** button cancels the freeze trace operation and resumes recording in the trace buffer. For additional information on using this button, see ***Freeze Trace Operation.***

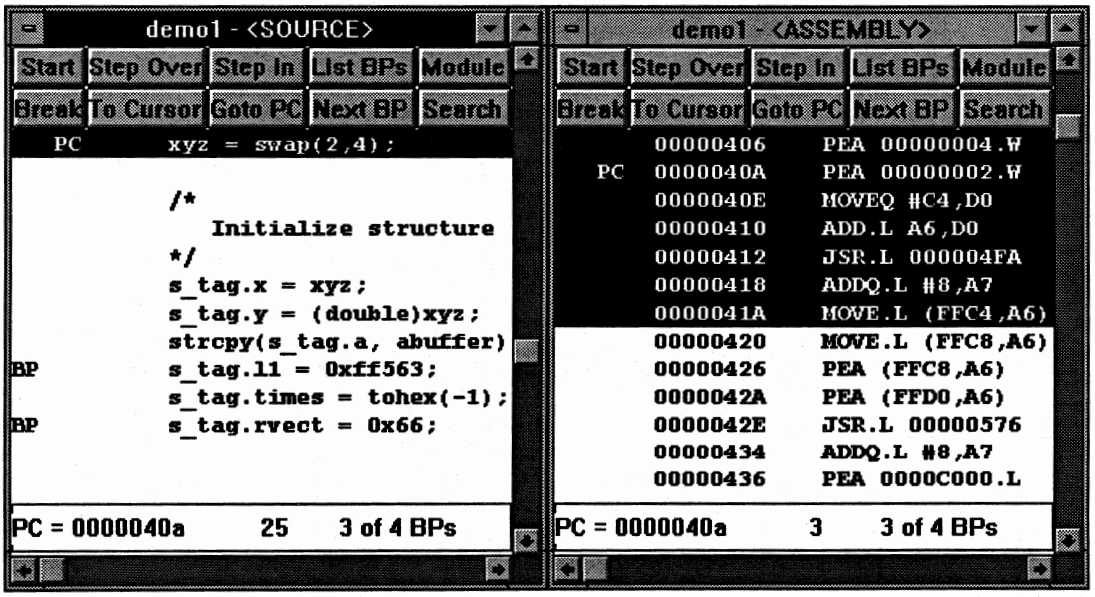
4 The Source Code Interface

When debugging source or assembly code, the Source Code Interface features of SourceGate II will prove to be invaluable to you. There are Code windows that allow you to perform emulator operations directly from your source file. There is also a Single Step window were various stepping scenarios can be performed along with a Go From/To window where you can define a source line to begin emulation from and source lines where emulation will stop. Each of these windows will be explained in detail in the sections that follow. **Note: The Source Code Interface examples shown in the following sections were obtained from a 68331/332 *Lite* emulator. 68HC16 *Lite* screens may show different data.**

Using The Code Window

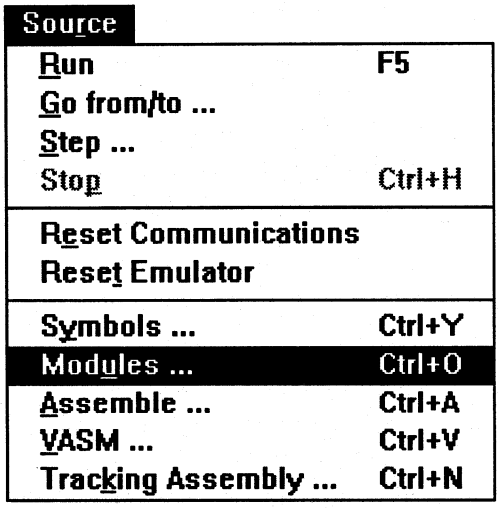
From inside of the Code windows, you can start and stop emulation, set breakpoints in your code, step over and into modules and subroutines, search your code for specific strings, jump to the current program counter location, list all set breakpoints, and open a different module to work with. Additional information is also shown, such as the number of breakpoints set, the current program counter value, and what lines of assembly are associated with the highlighted source line. Essentially, the entire operation of the emulator can be controlled from your source file.

An example of the Code window follows showing a source code module in one window with its corresponding assembly information shown in the assembly window next to it. Notice that the highlighted source line in the source window has the corresponding assembly for that line highlighted in the assembly window:

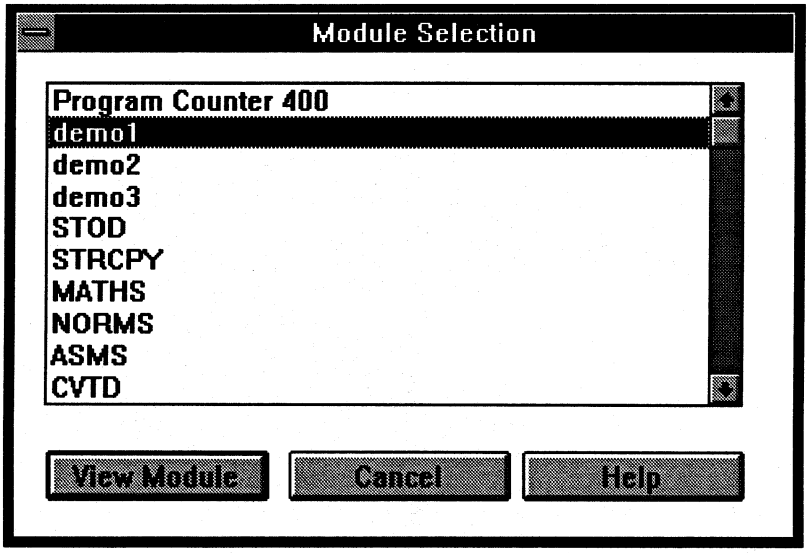


Opening A Code Window

To open a Code window, you must first load your source file using the 'File/Download Hex File' menu. Be sure to load a file that has been compiled with source level information. This information will be required to used the Source Code Interface. See your compiler documentation for additional information about creating an output file containing source level information. Once this file has been downloaded, you should choose 'Source/Modules':



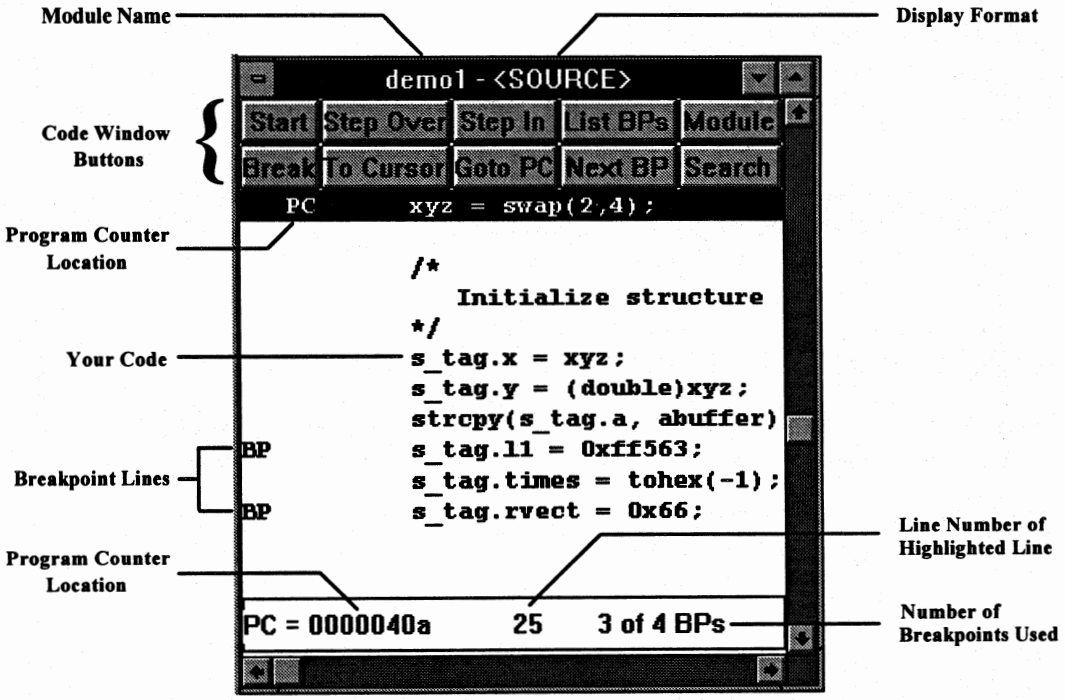
or use the accelerator key combination Ctrl+O. This will open the Module Selection window which allows you to view the modules that were loaded as part of your download file:



You can now highlight the module you wish to load into a Code window and then press the **View Module** button. A Code window will then be opened which contains the code for the selected module. To cancel the Module Selection operation, press the **Cancel** button. The **Help** button will invoke help for this window.

Code Window Format

The Code window was designed so that the maximum amount of information could be shown without the window becoming too cluttered. In the top border of the Code window, the module that you are viewing is shown followed by <SOURCE> or <ASSEMBLY>, depending on which display format you have selected. Directly below this information are two rows of buttons that allow you to perform certain functions. An overview of these buttons and the functions that they perform is provided in the sections that follow. The actual code information begins below the buttons. Any breakpoints that are set are designated by a "BP" label to the left of the line defined as a breakpoint. If "PC" is shown to the left of a line, that line is where the program counter currently resides. The bottom of this window shows the current program counter (PC) value, the line number of the currently highlighted line, and status information telling how many of the four available breakpoints are set. An example of a typical Code window is shown as follows:



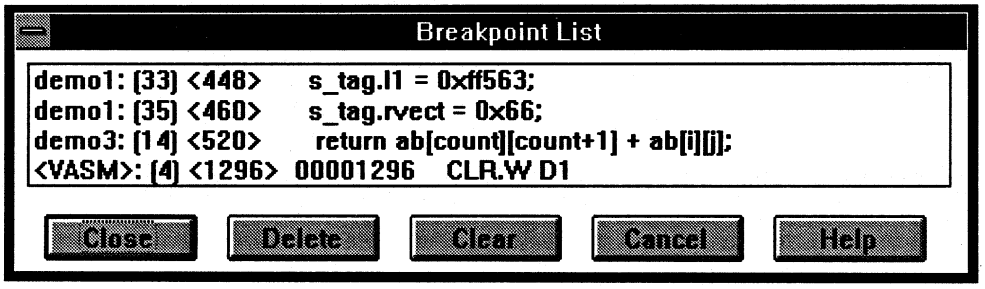
While you are performing stepping operations, the line containing the PC will be the line that is highlighted. This line will move as stepping proceeds into your code, opening up new source or assembly windows as needed to display where the PC is located. To select a line in the Code window, simply move the cursor to the desired line and click on it. This line will then become highlighted. Once a line has been highlighted, operations can be performed on it. These operations are reviewed in the sections that follow.

Setting Code Window Breakpoints

Setting a breakpoint on a line in a Code window will stop emulation when that line is encountered. Up to four breakpoints can be set in multiple Code windows. To set a breakpoint in a Code window, highlight the line you wish to set a breakpoint on by clicking on it. Then, once the line is highlighted, press the **Break** button. The highlighted line will then be labeled with "BP" showing that a breakpoint is set on that line. In addition, the bottom of all of the open Code windows will show "X of 4 BPs", where "X" is the number of breakpoints that have been set. If you try to set more than four breakpoints, the Delete Breakpoint window will appear and you must first delete a breakpoint before your new breakpoint can be set. **Note: For SourceGate II to find breakpoints set in the Code window, emulation must be started from the Code window by pressing the Start button.**

Deleting Code Window Breakpoints

Deleting breakpoints set in the Code window can be performed in one of two ways. Probably the easiest way is to highlight the line containing the breakpoint you wish to remove and then press the **Break** button. The **Break** button will toggle the breakpoint on and off for the highlighted line. The second method of deleting breakpoints in the Code window is through the use of the Breakpoint List window. This window lists all breakpoints set in the Code windows and is invoked by pressing the **List BPs** button. An example Breakpoint List window is shown as follows:



This window enables you to view the current set breakpoints and delete any of those not needed. Breakpoints are listed in the format:

module:[line#]<hex address> source line

where:

module — is the module that the breakpoint resides in. If the breakpoint is located in an assembly area that has no source line associated with it, the module will be listed as <VASM>.

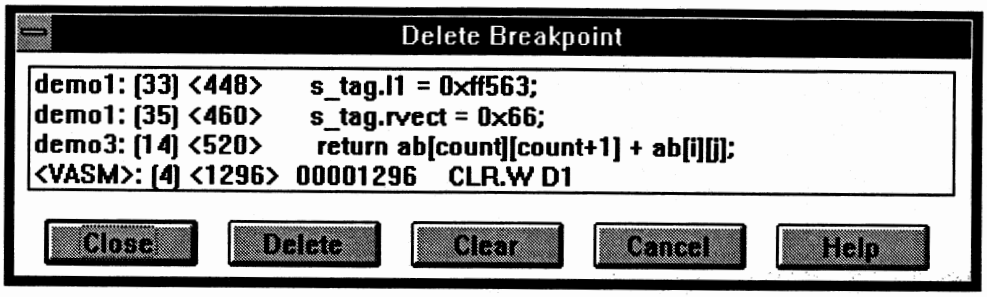
[line#] — is the line number of the module where the breakpoint is set. If <VASM> is shown as the module name, this number is the number of lines from the top of the VASM display to where the breakpoint is set.

<hex address> is the hexadecimal address of the line where the breakpoint is set.

source line — is the actual source or assembly listing of the breakpoint line.

To delete a breakpoint using this method, choose the breakpoint by clicking on it. This will highlight the selected breakpoint. The breakpoint can then be deleted by pressing the **Delete** button. All set breakpoints can be cleared by pressing the **Clear** button.

It is important to note that, if you attempt to set more than four breakpoints in the Code window, a window named Delete Breakpoint will appear:

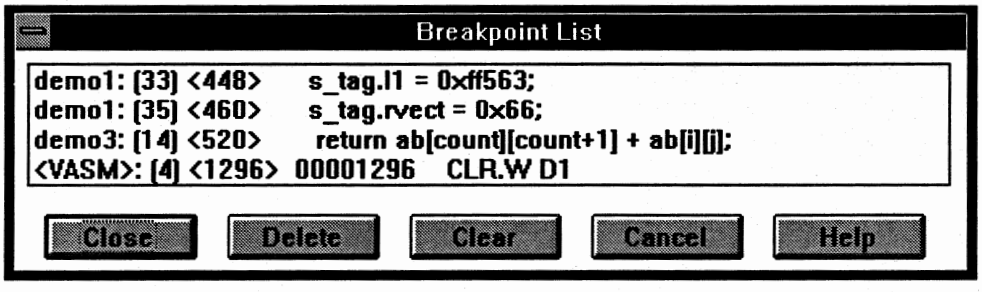


This window works exactly like the Breakpoint List window and is shown to you automatically so that you can delete a currently defined breakpoint to make room for your new breakpoint. Once you have selected/deleted a breakpoint, pressing the **Close** button will define your new breakpoint at the location you specified.

For information on the buttons available in the Breakpoint List and Delete Breakpoint windows, see ***Breakpoint List/Delete Breakpoint Window Buttons***.

Listing Code Window Breakpoints

To list the breakpoints set in a Code window, press the **List BPs** button. This will open the Breakpoint List window. This window lists all breakpoints set in the Code windows. An example Breakpoint List window is shown as follows:



This window enables you to view the current set breakpoints and delete any of those not needed. For additional information on how to use this window, see ***Deleting Code Window Breakpoints***. For information on the buttons available in the Breakpoint List window, see ***Breakpoint List/ Delete Breakpoint Window Buttons***.

Breakpoint List/Delete Breakpoint Window Buttons

A description of the functions of the buttons available in these windows are as follows:

Close

The **Close** button will close these windows. If you are using the Delete Breakpoint window and have deleted a breakpoint, pressing the **Close** button will add your new breakpoint to the list.

Delete

The **Delete** button will delete the highlighted breakpoint.

Clear

The **Clear** button will clear all currently defined breakpoints.

Cancel

The **Cancel** button will restore all breakpoints that have been deleted and will close this window.

Help

The **Help** button will invoke help for this window.

Finding Subsequent Code Window Breakpoints

To find subsequent Code window breakpoints, press the **Next BP** button. This will automatically place the highlighted line at the next line of code that is defined to be a breakpoint. This button will find breakpoints in the order that they were set. For example, if you cleared all existing breakpoints, then set breakpoints on line 5 of module A, line 8 of module B, and line 7 of module C, in that order, pressing the **Next BP** button will take you each of these breakpoints in order. However, the first breakpoint found depends on where you are when you press the **Next BP** button. If the highlighted line is on line 6 of module B when you press the **Next BP** button, the highlighted line will move to line 8 of module B (the second breakpoint set), another press of the **Next BP** button will take you to line 7 of module C (the third breakpoint set), and another press of the **Next BP** button will take you to line 5 of module A (the first breakpoint set). As you can see, the breakpoint that you go to when you press this button depends on where you start and the order in which the breakpoints were set.

Finding The Current Program Counter While In A Code Window

To find the current program counter (PC) location while in a Code window, press the **Goto PC** button. This will take the highlighted line to the Code window and line containing the PC. If the PC is not in a Code window that is open, and the PC points to a line of source code, the module containing the line pointed to by the PC will be loaded into a Code window and the line containing the PC will be highlighted. If the PC points to a line of assembly code, an assembly window will be opened beginning with the address of the PC.

Running To A Highlighted Code Window Line

Say you want to begin emulation from the current program counter location and run until a certain line in your code is reached. This can be done easily in a Code window. To do this, open a source or assembly Code window containing the line you wish to run to. Then highlight the line by clicking on it. When you press the **To Cursor** button, emulation will start and will continue until the highlighted line is reached. Care must be taken, however, to insure that the highlighted line will indeed be reached by your code or emulation will continue unabated.

Single Stepping In A Code Window

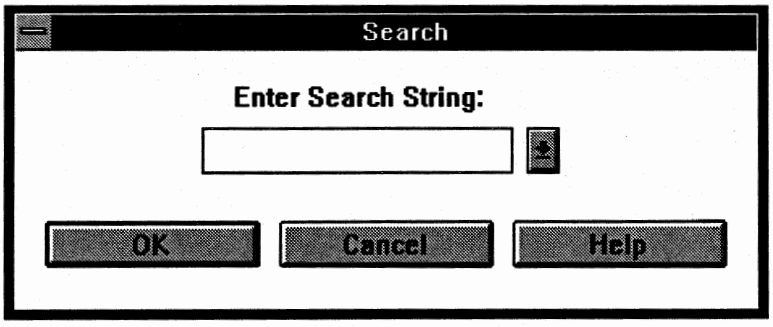
You can perform two different types of single stepping operations while in a Code window, Step Over and Step In.

Pressing the **Step Over** button will begin stepping from the current program counter (PC) location and will step one line of source code if in a source window and will step one line of assembly code if in an assembly window. Using this button **will not** allow the stepping to proceed into any other modules or assembly language routines. The code in these routines is actually executed, but the stepping process does not show this as stepping always appears to be in the active Code window. It should be noted that, if the code being stepped over is large, an hourglass will appear showing you that a step over operation is being performed. This hourglass will not appear if the cursor is pointing to a Code window button.

Pressing the **Step In** button will begin stepping from the current PC location and will step one line of source code if in a source window and will step one line of assembly code if in an assembly window. Using this button **will** allow stepping to proceed into any other module or assembly language routine. When a call or jump occurs in this mode, the source module or assembly rountine is automatically loaded into a Code window in SOURCE or ASSEMBLY mode (respectively), and stepping proceeds in this new window.

Searching For A Code Window String

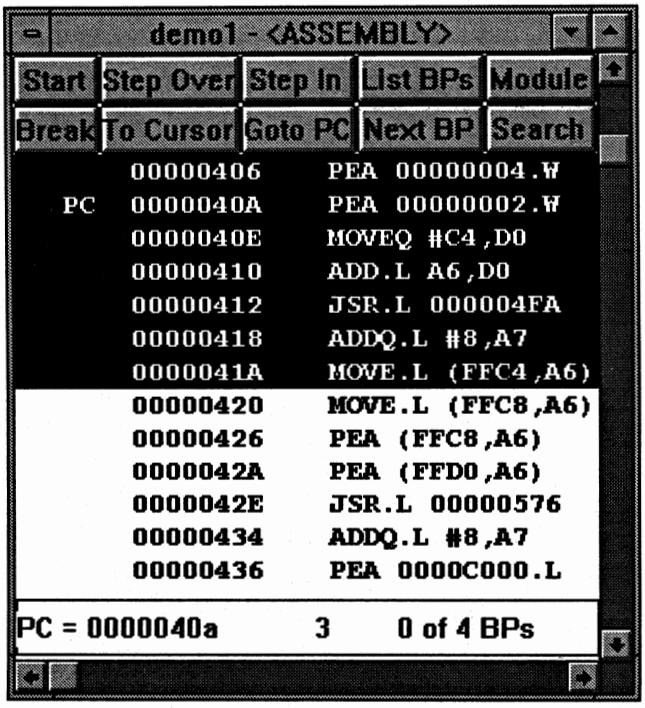
You can search a Code window for a string. This can be useful when looking for a comment or a specific variable name. To begin a search, press the **Search** button. A Search window will be shown as follows:



Enter the string to be searched for and press the **Ok** button to begin the search. Previously entered strings can be recalled by pressing the **Scroll Box** button. Use the **Cancel** button to cancel the search operation. The **Help** button will invoke help for this window.

Viewing Code Window Assembly

If you are working with a Code window that contains source information, you can show the assembly information for that module also. This assembly window will track the source window it was invoked from . The highlighted lines in the assembly window correspond to the highlighted lines in the source window. An example of an assembly window is shown as follows:

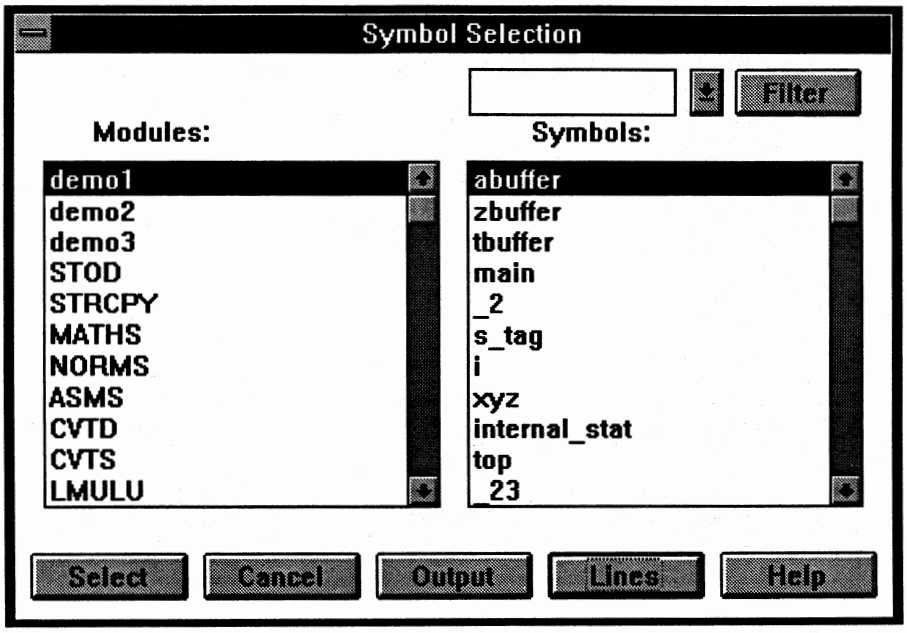


To open the assembly window for a module, you must have a Code window opened that contains a source module. Then, choose 'Source/Assemble' or you can use the accelerator key combination Ctrl+A. The corresponding assembly Code window will be displayed.

Once the assembly window is open, you can make it the active window and use its buttons to perform commands. When you do this, the corresponding source window will then track the assembly window. See the first page of this chapter for an example of a source window and its corresponding assembly window being shown together.

Viewing Code Window Symbols

If you need to see what symbols are defined for a particular module, you can open the Symbol Selection window. An example Symbol Selection window is shown as follows:



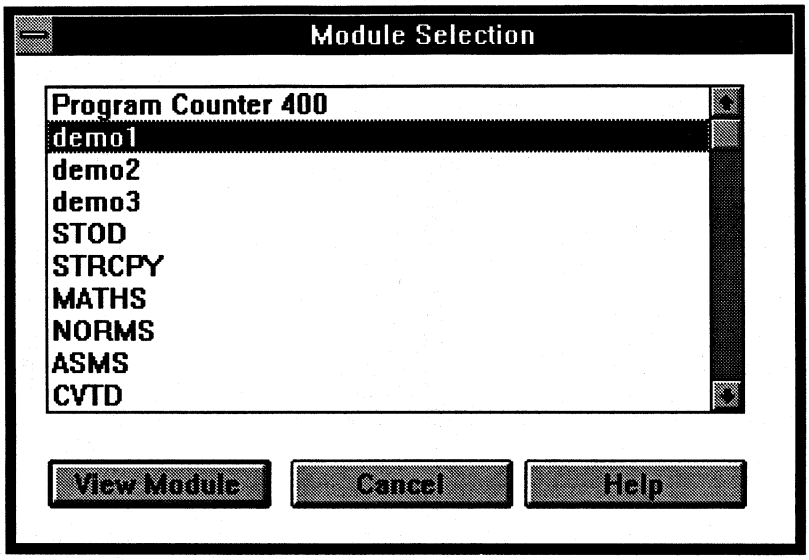
To open this window, choose 'Source/Symbols'. An alternate method of opening the Symbol Selection window is to use the accelerator key combination Ctrl+Y.

The symbols available for a module can by viewed by clicking on the desired module. There is a **Lines** button that will switch the display to the line number symbols available for the module. When line numbers are shown, this button will switch back to **Symbols**. When this window is invoked from a Code window, the **Select** button is not functional.

For additional information on using the Symbol Selection window, see *Working With Symbols* in Chapter 2.

Selecting Other Modules From A Code Window

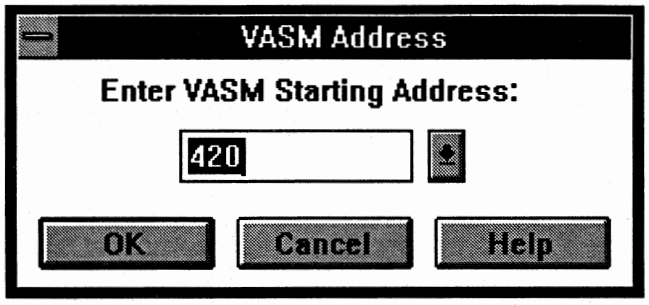
You can select other modules from a Code window by pressing the **Module** button or by using the accelerator key combination Ctrl+O. This will open the Module Selection window which allows you to view the modules that were loaded as part of your download file:



You can now highlight the module you wish to load into a Code window and then press the **View Module** button. A Code window will then be opened which contains the code for the selected module. To cancel the Module Selection operation, press the **Cancel** button. The **Help** button will invoke help for this window.

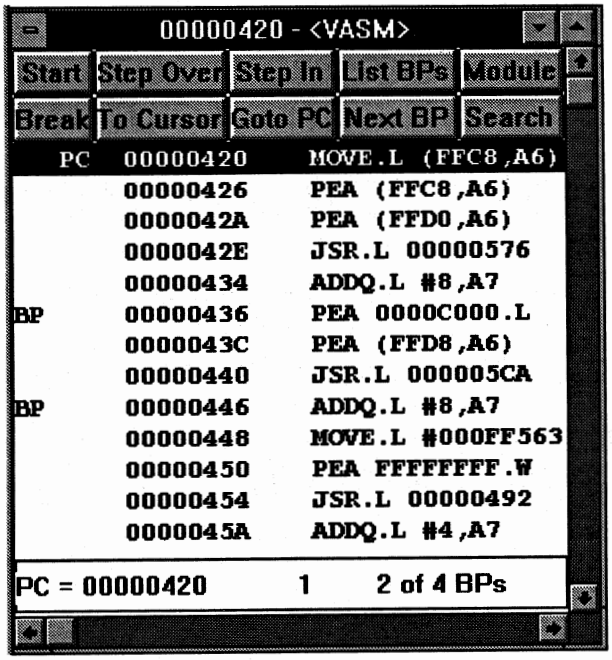
Using The VASM Window

The View Assembly (VASM) window allows you to load a Code window with assembly information beginning at an address you specify. All of the functions of a normal Code window are available in this VASM window. To open a VASM window, choose 'Source/VASM' or use the accelerator key combination Ctrl+V. You will then be prompted with a VASM Address window:



Enter the address from which you wish to start the VASM window in the field provided and then press the **Ok** button. A VASM window will then be opened beginning at this address. Previously entered VASM addresses can be recalled by pressing the **Scroll Box** button. Use the **Cancel** button to cancel the VASM address operation. The **Help** button will invoke help for this window.

An example VASM window that was generated from the above address entry is shown as follows:

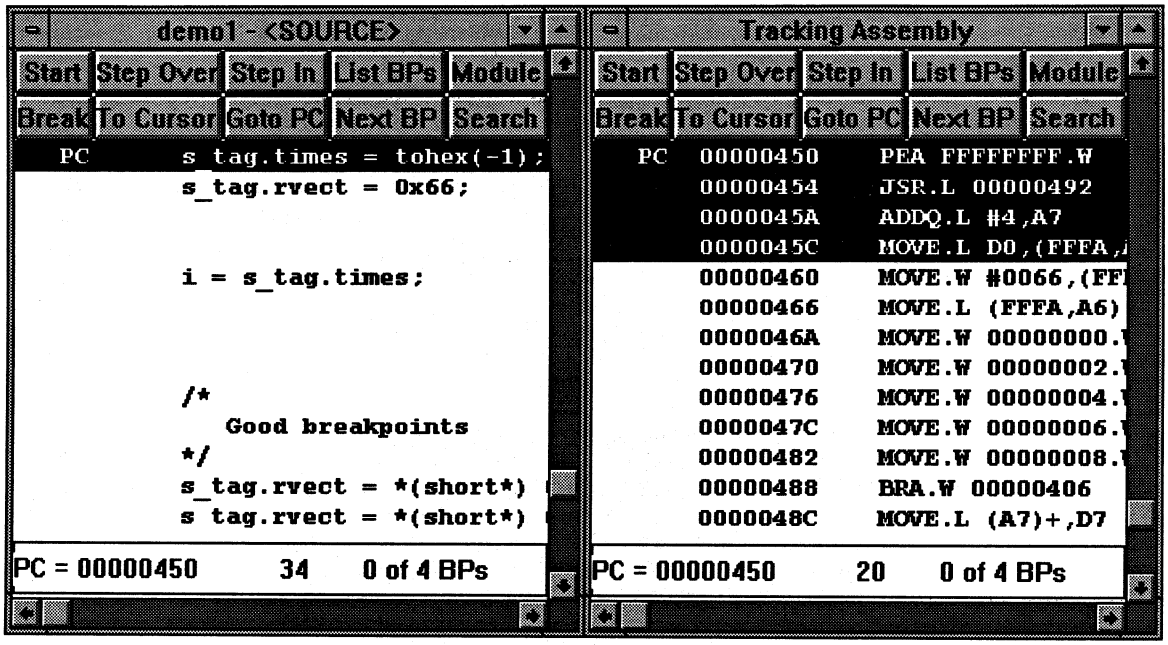


Since this window is just a specialized version of a Code window, see *Using The Code Window* for additional information on using this window.

Using The Tracking Assembly Window

The Tracking Assembly window is an assembly window which will remain on the screen and will track the source window that currently has the focus. This is done by highlighting the assembly lines that correspond to the source line highlighted in the active module. If this window is activated when no source modules are open, it will show assembly code beginning at the current program counter location.

The Tracking Assembly window can be opened by choosing 'Source/Tracking Assembly'. An example Tracking Assembly window,along with the source window that it is tracking, are shown as follows:

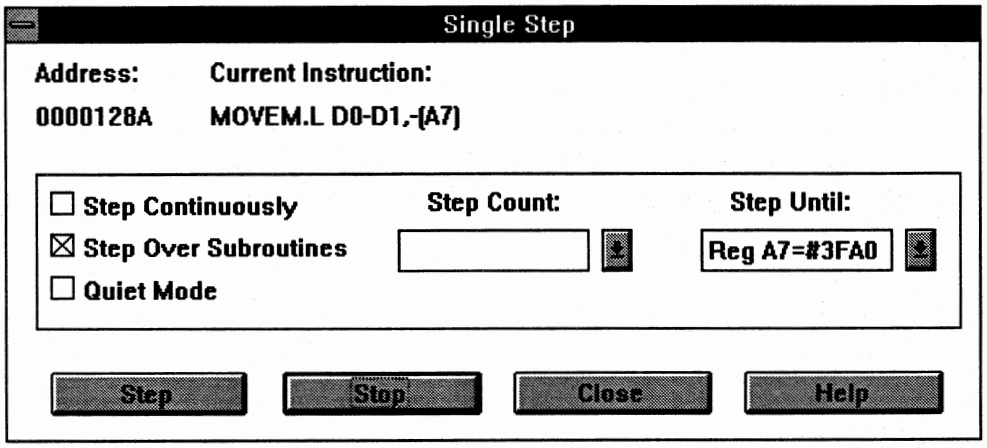


This window can be very useful as you can open it, position it on the screen where you want it, and it will always track whatever source Code window you are currently working in. When you go into assembly, it will mirror the assembly window you are working with.

Since this window is just a specialized version of a Code window, see *Using The Code Window* for additional information on using this window.

Using The Single Step Window

If you prefer to do your single-stepping outside of a Code window, the Single Step window will allow you to do this. This window allows you to single-step at the assembly level. Several conditions define the stepping operation including using step counts and step until conditions to terminate the stepping operation. To open the Single Step window, choose 'Source/Step'. An example Single Step window is shown as follows:



Single Step Window Fields

The Single Step window has several fields that determine how single-stepping occurs. An overview of these fields is given in the sections that follow.

Address Field

The 'Address' field shows the location of the program counter. This is the address that will be executed next when single stepping is performed.

Current Instruction Field

The 'Current Instruction' field shows the instruction pointed to by the program counter. This is the next instruction that will be executed when single stepping is performed.

Step Continuously Check Box

The 'Step Continuously' box, when checked, will allow stepping to proceed continously until the **Stop** button is pressed. If this option is not checked, one instruction will be executed each time the **Step** button is pressed.

Step Over Subroutines Check Box

The 'Step Over Subroutines' box, when checked, will NOT allow stepping to proceed into any called routines. If this option is not checked, stepping will proceed into the called routine.

Quiet Mode Check Box

The 'Quiet' box, when checked, will only show the first instruction executed and will not update the 'Address' and 'Current Instruction' fields until the **Stop** button is pressed. At that point, the last instruction executed will be shown.

Step Count Field

The 'Step Count' field allows you to enter the number of steps desired. When the **Step** button is pressed, the entered number of steps will occur and then stepping will stop. Notice that, when a value is entered in this field, the 'Step Continously' option is disabled. To enter a value in this field, place the mouse cursor in this field and press the left mouse button. The **Scroll Box** button can be used to view and select any previously entered step counts.

Step Until Field

The 'Step Until' field allows you to enter a certain condition on which stepping will stop. When the **Step** button is pressed, stepping will begin and will stop when the specified condition is encountered. To enter a value in this field, place the mouse cursor in this field and press the left mouse button. The **Scroll Box** button can be used to view and select any previously entered step conditions.

The Step Until clause is defined as follows:

```
OPERAND1 OPERATION OPERAND2
```

Where OPERAND1 can be:

```
     Reg <reg> - Register name
Long at <addr> - Long word value at address <addr>
Word at <addr> - Word value at address <addr>
Byte at <addr> - Byte value at address <addr>
```

OPERAND2 can be:

```
 Reg <reg> - Register name
  #<value> - Immediate value (size determined by OPERAND 1)
    <addr> - Value at <addr> (size determined by OPERAND 1)
```

OPERATION can be:

```
 = - OPERAND1 equal to OPERAND2
== - OPERAND1 equal to OPERAND2
!= - OPERAND1 not equal to OPERAND2
<> - OPERAND1 not equal to OPERAND2
>< - OPERAND1 not equal to OPERAND2
 > - OPERAND1 greater than OPERAND2
 < - OPERAND1 less than OPERAND2
>= - OPERAND1 greater than or equal to OPERAND2
=> - OPERAND1 greater than or equal to OPERAND2
<= - OPERAND1 less than or equal to OPERAND2
=< - OPERAND1 less than or equal to OPERAND2
```

Example:

SS Until word at 5000 = #1234

Would stop stepping when 1234H appears at address 5000H. It should be noted that this command takes into account the way data is stored in memory based on the microprocessor manufacturer. If, for example, you wished to stop on a word value of 1234H being in location 5000H, the emulator will look for a 1234H at location 5000H for Motorola processors and 3412H at location 5000H for Intel, Hitachi, and Zilog processors.

Single Step Window Buttons

A description of the functions of the buttons available in this window follows:

Step

The **Step** button will begin stepping if 'Step Continuously' is checked, and will step one instruction each time that the **Step** button is pressed if 'Step Continuously' is not checked.

Stop

The **Stop** button will stop the stepping operation.

Close

The **Close** button will close the Single Step window.

Help

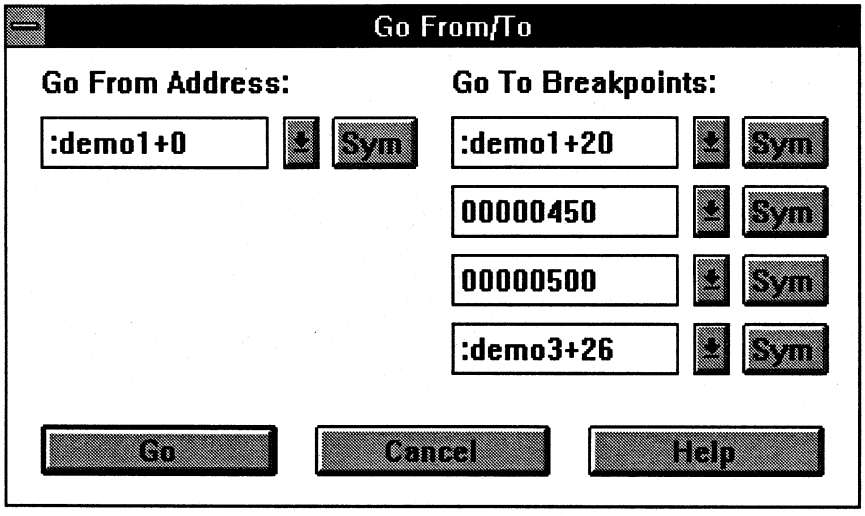
The **Help** button will invoke help for this window.

Scroll Box

Use the **Scroll Box** button to select from previously entered values.

Using The Go From/To Window

The Go From/To window allows you to begin emulation from a specific address and, once emulation is begun, break on up to four defined breakpoints. This can be handy when you need a down and dirty way to set temporary breakpoints without going through the event system. To open the Go From/To window, choose 'Source/Go From/To'. An example Go From/To window is shown as follows:



Go From/To Window Fields

The Go From/To window has several fields that determine how this window functions. An overview of these fields is given in the sections that follow.

Go From Address Field

The 'Go From Address' field defines the address or line number where emulation will begin when the **Go** button is pressed. A hexadecimal address can be entered here or a line number of a module can be selected by pressing the **Sym** button. The **Scroll Box** button can be used to select from previously entered values.

Go To Breakpoints Fields

The 'Go To Breakpoints' fields define the addresses or line numbers where emulation will stop when any of them are encountered. The four breakpoints defined here are evaluated as a logical OR condition where emulation will stop when any of these conditions occurs. A hexadecimal address can be entered here or a line number of a module can be selected by pressing the **Sym** button. The **Scroll Box** button can be used to select from previously entered values.

Go From/To Window Buttons

A description of the functions of the buttons available in this window follows:

Sym

The **Sym** button will open the Symbol Selection window where you can select a module and line number to be put in the field. **For additional information on using the Symbol Selection window, see *Working With Symbols* in Chapter 2.**

Go

The **Go** button will begin emulation from the defined address. Emulation will stop when the first of the four breakpoint conditions occurs.

Cancel

The **Cancel** button will cancel the defined Go From/To setup and will close the Go From/To window.

Help

The **Help** button will invoke help for this window.

Scroll Box

Use the **Scroll Box** button to select from previously entered values.

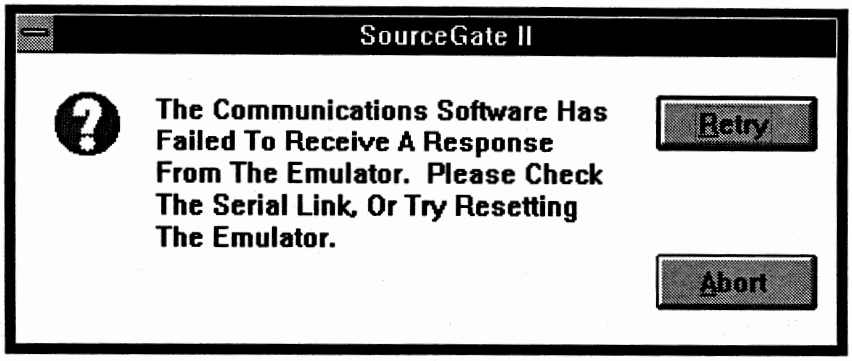
APPENDICES

Appendix A Error Messages 178
Appendix B Using Compilers and Assemblers 183

Appendix A Error Messages

The following is an overview of some of the more common error messages that could be encountered when using SourceGate II. If you encounter an error message that is not listed here, contact HMI Technical Support.

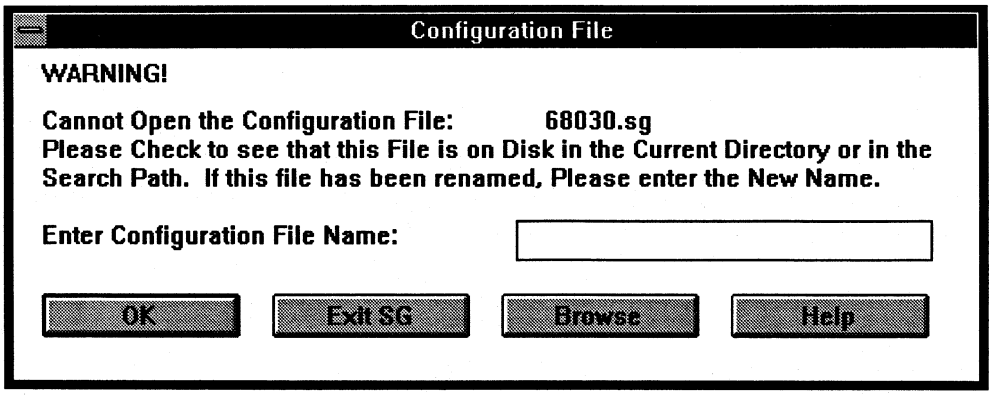
ERROR:


SourceGate II
The Communications Software Has Failed To Receive A Response From The Emulator. Please Check The Serial Link, Or Try Resetting The Emulator.
Retry
Abort

CAUSE:
SourceGate II could not establish communications with the emulator.

SOLUTION:
This problem can be caused by several conditions. First, make sure that the emulator is powered on. The green LED on the front of the emulator should be on. If it is not, check the power connection to the emulator. Second, make sure that a straight-through type of RS-232 cable is plugged into the RS-232 port of the emulator on one end, and to a valid serial port on the computer end. Third, make sure that SourceGate II is being invoked with the proper COM port setting as outlined in the section ***Starting SourceGate II***. After making the above inspections, press the **Retry** button. If you still have problems after trying the above fixes, call Technical Support.

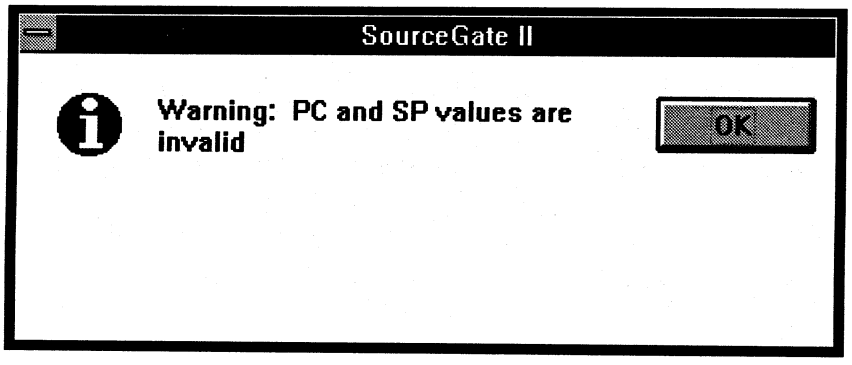
ERROR:


Configuration File
WARNING!
Cannot Open the Configuration File: 68030.sg
Please Check to see that this File is on Disk in the Current Directory or in the Search Path. If this file has been renamed, Please enter the New Name.
Enter Configuration File Name:
OK
Exit SG
Browse
Help

CAUSE:
The personality file for your particular emulator could not be found in the current directory or in the search path.

SOLUTION:
Use the **Browse** button to find the correct drive/directory and .SG filename. You can also move the named file to your current directory, change to the directory where this file resides, or include the directory where this file resides in your search path. Then, press the **OK** button.

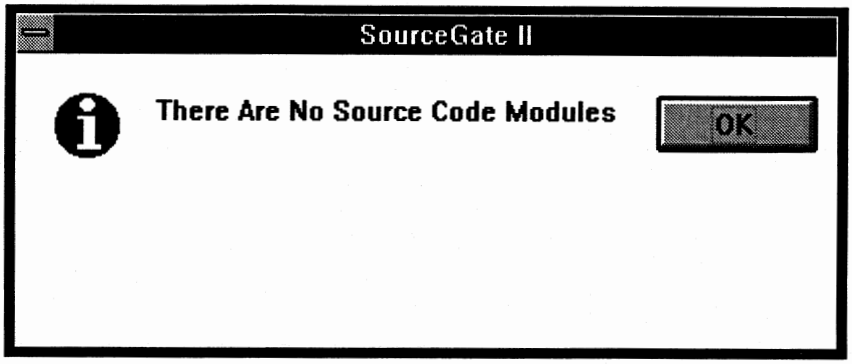
ERROR:



CAUSE:
Certain emulators, most notably the CPU32 family, fetch their reset and stack pointer vectors from memory upon power-up. If this memory is mapped to the emulator, the values fetched for these vectors will be random garbage since the processor is fetching values from uninitialized RAM.

SOLUTION:
Edit memory locations 0 - 7 so that they contain valid reset and stack pointer vectors and then perform the "RS" command from the Command window, or press the "RESET" button in the Run Status window.

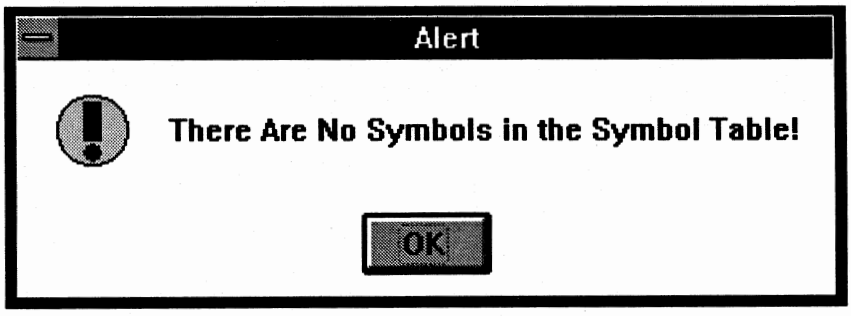
ERROR:



CAUSE:
You attempted to perform a source-level debugging operation. However, a file containing source-level debugging information has not yet been loaded.

SOLUTION:
Load a file that contains source-level debugging information and try again.

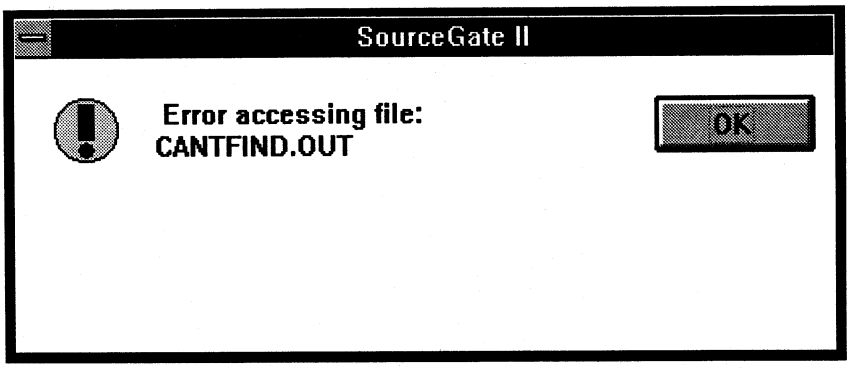
ERROR:



CAUSE:
You attempted to perform an operation that referenced a symbol. However, a file containing symbolic debugging information has not yet been loaded.

SOLUTION:
Load a file that contains symbolic debugging information and try again.

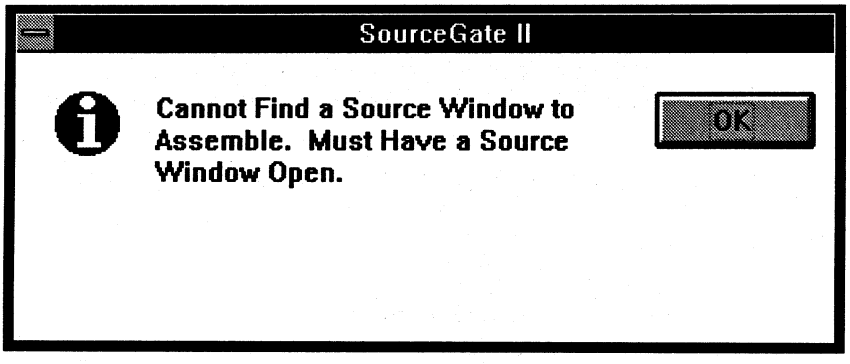
ERROR:



CAUSE:
SourceGate II cannot find the file specified.

SOLUTION:
Make sure that the filename and path specified are correct.

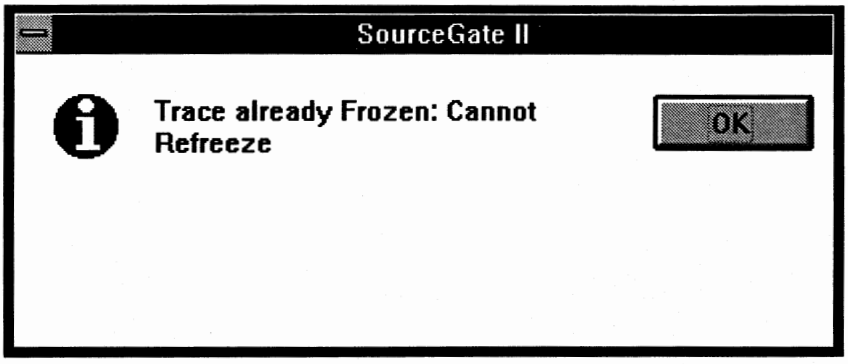
ERROR:



CAUSE:
You tried to invoke an assembly window when no source modules are active.

SOLUTION:
Open a source window and then invoke an assembly window, or use the VASM command to open an assembly window. The VASM command allows you to open an assembly window at any time beginning at the defined address.

ERROR:



CAUSE:
You tried to freeze the trace after the emulator was already in Freeze Trace mode.

SOLUTION:
This is not really an error condition but merely a way to alert you that you are already in Freeze Trace mode. If you want to freeze the trace again, you must first Resume the trace and then another Freeze Trace operation can be performed.

Appendix B Using Compilers and Assemblers

SourceGate is compatible with a number of downloadable file formats. SourceGate can load the code, data, symbolic, and source-level debugging information (line numbers) from those formats that supply such information.

SourceGate supports the following formats (those that do not produce symbol information (including line numbers) are so noted):

Motorola S-Record (No symbols)
Intel Hex (No symbols)
Tekhex (No symbols)
HMI Binary (No symbols)
Common Object File Format (COFF)
Software Development Systems (SDS)
Archimedes Debug (IAR)
IEEE-695
Extended Tekhex
Microtec Research Absolute
Hitachi

Most compiler/assembler manufacturers provide linkers that can produce one of the above formats. Other formats can be converted to Extended Tekhex with object file convertors supplied by HMI. These converters are available through the HMI-BBS at 205-881-7395, 8-N-1, 24 hours a day, 7 days a week. If you don't have access to a modem, call us and we'll send you a converters disk. All HMI converters supplied on the converters disk are outlined in the following paragraphs.

The HMI converters disk contains executable (.EXE or .COM) and documentation (.DOC) files for each converter listed in the following sections. Please read the applicable documentation file for the converter you wish to use **before** running the converter. The documentation files have been written to provide insight on how to use the converter programs properly.

NOTE: Chances are you will not need any of these converters in order to read your object files correctly. If you have a doubt about whether or not your output format is supported directly, try reading the file directly into SourceGate II. If it is a valid format, it will be read correctly. If not, you will get an error message. You can then use one of the converters to convert the file into one of the above formats. If you have any questions about what type of output file your compiler/assembler produces, call your software supplier or HMI Technical Support.

INDEX TO THE FILE CONVERTERS DISK

180AD2XT	-	2500 A.D. 64180 symbol table file to XTEKHEX file converter.
2500TOXT	-	2500 A.D. standard symbol table to Extended TEKHEX file converter.
68KAD2XT	-	2500 A.D. MC68000 symbol table file to Extended TEKHEX file converter
AATOXT	-	American Automation hex file to Extended TEKHEX file converter.
ATOXT	-	UNIX A.OUT file to Extended TEKHEX file converter.
AVOXTEK	-	AVOCET plain dump symbol file to Extended TEKHEX file converter.
DBGXTEK	-	Manx debug (DBG) file to Extended TEKHEX file converter.
HTOXT	-	HITACHI hex file to Extended TEKHEX file converter.
ITOB	-	INTEL hex file to BINARY file converter.
ITOM	-	INTEL hex file to MOTOROLA 'S' record file converter.
MTTOXT	-	MICROTEK symbol table to Extended TEKHEX file converter.
OBJCNV	-	INTEL Object file converter. (The Intel Object file converter operates on an Intel object format file produced by the Intel programs LOCATE, RL51, and/or RL96. It produces as its output a standard Intel format HEX file, and a standard HMI format symbol file.)
QUEXTEK	-	QUELO linker report file to Extended TEKHEX file converter.
UTILXTEK	-	Intermetrics (Whitesmiths) Object file to Extended TEKHEX file converter.
ZTOXT	-	INTEL hex file/ZILOG symbol file to Extended TEKHEX file converter.

INDEX TO THE FILE CONVERTERS DISK (Cont.)

IAR2XT	-	IAR (Archemedies) UBROF Debug file to Extended TEKHEX file converter.
COFF2XT	-	COFF file (Introl, Sierra, Intermetrics, SUN386, and PG52x0 COFF) to Extended TEKHEX file converter.
IEEETOXT	-	IEEE 695 Object format file to Extended TEKHEX file converter.
IEEE2IAR	-	IEEE 695 Object format file to Archimedes Debug (IAR) file converter.
NWISTOXT	-	Northwest Instruments Symbol file to Extended TEKHEX file converter.
COMFORXT	-	COMFOR file to Extended TEKHEX file converter.
ARCH2BNK	-	Archimedes Banked "S" Record Splitter
OS92HMI	-	Microware OS2 Format To SDS Format File Converter

Index

SYMBOLS

:, 43
:=A, 44
:=C, 44
:=D, 44
:=M, 44
:=S, 45
:module/symbol, 43
:module/symbol=value, 43
:symbol, 43
:symbol*, 43
:symbol=value, 43

A

A (In-Line Assembly), 48
Accelerator Keys, 24
Adding A Watch Variable, 128
Address Registers A0-A7, 141
Alternate Function Code Registers, 142
Assembly Mode Command, 44
Assembly Window, 10
Automating SourceGate II, 23

B

B (Byte Mode), 41
Begin Emulation, 9
BM (Byte Mode), 49
Breakpoint, 8
BSD(+/-) (Tri-State Data Bus Buffers During DMA), 50
BTE (Set 8-Bit Boot Chip Select), 51

C

C (Compare Memory Block), 52
Checking Your Package, 1
Clear Symbol Table Command, 44
CM (Show Status Of Control Board Memory), 54
Code Window, 160
Code Window Format, 162
Command File, 7
Command file, 8
Command Line Options, 3
Command Line Overrides, 41
Command Window, 8, 39
Comparing Memory Blocks, 144
CONFIG (Recall Factory Configuration), 55
CONFIGT (Recall Factory Configuration For Target), 56
Configuration Window, 25
Control Menu Boxes, 6
CRL (Load Configuration Registers), 57

D

D (Data Mode), 41
D (Dump Memory Block), 58
Data Registers D0-D7, 141
DDP (Dump Dual Port Memory Block), 59
Defining Breakpoints, 108
Defining Trace Triggers, 108
Deleting A Watch Variable, 134
Deleting Code Window Breakpoints, 163
Disk Space Requirements, 2
Display Current Modes Command, 44
DM (Data Memory Mode), 60
Download Hex File, 33
Download Hex File, 7
Download Options, 34

E

E/EN (Examine/Change Memory), 61
EDP/EDPN (Examine/Change Dual Port Memory), 62
Emulator To Computer Connection, 2
Emulator To Target System Connection, 2
Environment Variables, 5
Error Messages, 178
Event, 8, 104
Event Window, 104
Event Window Fields, 105
Exiting SourceGate II, 8, 22
EXP (Examine/Change Peripheral), 63
EXR (Examine/Change Peripheral Register), 64

F

F (Fill Memory Block), 65
FDP (Fill Dual Port Memory Block), 66
File Converters Disk, 184
Filling Memory With A Constant, 146
Finding Subsequent Code Window Breakpoints, 165
Finding The Current Program Counter While In A Code Window, 165
Freeze Trace, 119

G

G (Begin Real-Time Emulation), 67
Go From/To Window, 9, 175

H

Help System, 11, 17
HLD (Hex Listing Disabled), 68
HLE (Hex Listing Enabled), 69

I

I/IN (Input), 70
Iconizing A Window, 13
Installing SourceGate II, 2

L

L (List Disassembled Code), 71
L (Long Word Mode), 41
Listing Code Window Breakpoints, 164
LM (Long Word Display Mode), 72
Log Input/Output To File, 8

M

M (Move Memory Block), 73
Maximize Button, 7
Memory Compare Window, 144, 146
Memory Configuration, 25
Memory Goto Window, 147
Memory I/O Window, 148
Memory Map, 10, 25
Memory Move Window, 149
Memory Requirements, 2
Memory Search Window, 150
Memory Test Window, 152
Memory Window, 9, 143
Minimize Button, 7
Mixed Mode Command, 44
MMC (Check Configuration Menu Setup), 74
Modifying Memory Values, 144
Module Commands, 44

Index

Module Selection Window, 10, 161, 169
Moving Memory Blocks, 149
MT/MTL (Test Block Of Memory), 75

O

O/ON (Output), 76
Object file, 7
Object File Formats, 33
Opening A Code Window, 161
Override Example, 42

P

P (Program Mode), 41
PM (Program Memory Mode), 77
Printer Setup, 8
Printing The Trace Buffers, 125
Program Counter Description Field, 140
Program Counter Register, 140
PROM, 7, 37

R

R (Repeat), 42
RD(%) (Read Configuration File), 80
RD(!) (Read Firmware File Into Flash ROM), 81
RD/RDP (Read Object File Into Memory), 78
Reading A Specific Memory Location, 148
Real Time Watch Window, 138
Register Window, 9, 139
Register Window Contents, 140
Reset Communications, 10
Reset Emulator, 10
Restore Button, 7
Restoring A Saved Watch Window, 134
Resume Trace, 119
RP (Reset Processor, Continue Emulation), 82
RS (Reset Processor, Stop Emulation), 83
Run Status Window, 9, 158
Run Status Window Fields, 158
Running To A Highlighted Code Window Line, 166

S

S (Supervisor Mode), 41
Saving A Watch Window, 134
Scroll Bars (Horizontal and Vertical), 7
Scroll Boxes (Horizontal and Vertical), 7
Searching For A Code Window String, 166
Searching Memory Blocks, 150
Searching The Trace Buffers, 120
Selecting A Printer, 16
Selecting Other Modules From A Code Window, 169
Setting Code Window Breakpoints, 163
SH (Show Status For SHD/SHE), 84
Shadow RAM, 138
SHD (Serial Handshake Disabled), 85
SHE (Serial Handshake Enabled), 86
Single Step Window, 9, 172
Single Stepping In A Code Window, 166
SM (Supervisor Memory Mode), 87
Source Code Interface, 160
Source Code Interface Options, 10, 32
Source Mode Command, 45
SourceGate II Menu Bar, 6, 7
SourceGate II Title Bar, 7
SP (Stop Emulation), 88
SR/SRN (Search Block Of Memory), 89
SS (Single Step), 91
ST (Show Emulator Status), 94
Stack Trace Window, 9, 156
Starting SourceGate II, 2
Starting SourceGate II From File Manager, 2
Starting SourceGate II From Program Manager, 2
Status Register, 140
Stop Emulation, 9
Supervisor Stack Pointer Register, 141
Symbol Commands, 43
Symbol Selection Window, 10, 17, 133, 168
System Requirements, 1

T

T (Target System), 41
Testing Memory Blocks, 152
Trace Display Format, 112
Trace Search Window, 120
Trace Trigger, 8
Trace Window, 9, 110
Tracking Assembly Window, 10, 171

U

U (User Mode), 41
UM (User Memory Mode), 95
Upload Hex File, 7, 37
User Stack Pointer Register, 141
Using Compilers and Assemblers, 183

V

VASM (View Assembly), 96
VASM Window, 10, 170
Vector Base Register, 141
VER (Display Firmware Version), 97
Viewing A Specific Address In Memory, 147
Viewing Code Window Assembly, 167
Viewing Code Window Symbols, 168

W

W(%) (Write Configuration File), 100
W (Word Mode), 41
W (Write Memory Contents To File), 98
Watch In Real Time Window, 9
Watch Window, 9, 127
Watch Window Examples, 135
Window Manipulation, 11, 15
WM (Word Display Mode), 101
Writing A Specific Memory Location, 148

X

X (Examine/Change Registers), 102